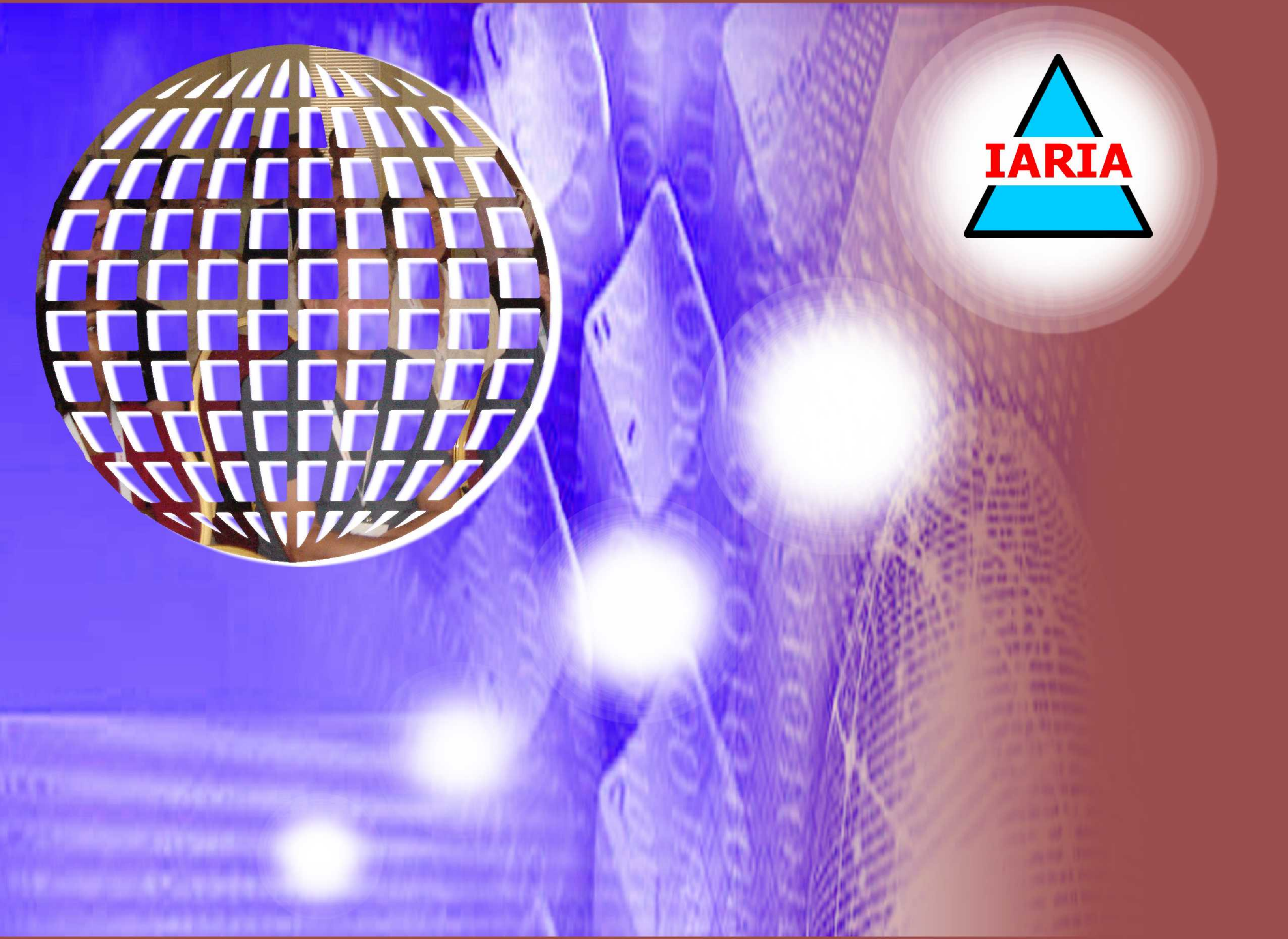
International Journal on Advances in Networks and Services



2014 vol. 7 nr. 3&4

The *International Journal on Advances in Networks and Services* is published by IARIA.
ISSN: 1942-2644
journals site: http://www.iariajournals.org
contact: petre@iaria.org

International Journal on Advances in Networks and Services
Volume 7, Number 3 & 4, 2014

International Journal on Advances in Networks and Services
Volume 7, Numbers 3 & 4, 2014

CONTENTS

Wireless Condition Monitoring for Industrial Applications based on Radio Sensor Nodes with Energy Harvesting

Michael Niedermayer[1], Stephan Bennecke[2], Rainer Wirth[3], Eduard Armbruster[1], Klaus-Dieter Lang[2]
[1]Dept. RF & Smart Sensor Systems, Fraunhofer IZM, Berlin, Germany, email: {niederm, armbruster}@izm.fraunhofer.de
[2]Dept. Center of Advanced Packaging, Techn. University of Berlin, Berlin, Germany, email: {benecke, tib421}@tu-berlin.de
[3]GfM Gesellschaft für Maschinendiagnose mbH, Berlin, Germany, email: wirth@maschinendiagnose.de

Abstract — **Wireless sensor nodes can detect component wear and tear just in time. Such sensor networks can allow for waiting to overhaul delay the overhauling of machines until it is really absolutely necessary. This means that time-consuming and expensive downtime can be avoided. A sensor network for the condition monitoring of industrial machines has been developed in the publicly funded project ECoMoS. The implemented sensor nodes are able to predict machine failures until three months in advance.**

Keywords — Energy Harvesting; Condition Monitoring; Machine Diagnosis; Wireless Sensor Networks

I. Introduction

With the availability of inexpensive, self-powered wireless sensor nodes the burden of condition monitoring systems drops significantly. The cost savings by introduction of wireless condition monitoring solutions can be considerable. Furthermore, new service concepts in conjunction with a maintenance cloud are enabled by a role-specific access to the sensor data and diagnostic results. Such cloud based sensor networks for condition monitoring will avoid time-consuming and expensive downtime of much industrial equipment in the near future. Sensor networks offer the chance to revolutionize measurement technology in the coming years. In the joint project 'Energy-autarkic Condition Monitoring System' (ECoMoS), the application field of self-powered wireless sensor technology has been expanded to the wireless condition monitoring of industrial plants [1]. A lot of research has been done to prepare this development. The preceeding research activities focused on the implementation of sensor nodes with small size [2][3] at minimal costs [4][5] for industrial enviroments [6][7].

The rest of the paper is organized as follows: first, we discuss relevant aspects of vibration diagnosis in Section II. Afterwards, prototypes for energy harvesting are presented in Section III. Section IV describes several implementation details of the wireless sensor network. The field test is introduced in Section V. Finally, Section VI concludes our work and gives a perspective on our future work.

II. Detecting machine damages in advance

Several measurement categories can be checked for the condition monitoring of machines. Temperature and vibration signals are most frequently analyzed to identify malfunctions of machines. Here, we focus on condition monitoring based on acceleration sensors, which measure vibrations. Vibration diagnosis was established for early prediction of machine breakdowns. The corresponding signal analysis methods are usually based on spectral analyses of the occurring mechanical vibrations. Two types of machine diagnosis can be distinguished regarding the accuracy. In one variant, the basic diagnosis utilizes methods of simple characteristics. The second variant of condition monitoring applies an in-depth diagnosis to predict machine failures several weeks in advance. The higher accuracy can be achieved with the help of diagnostic algorithms, which use knowledge of kinematic drive models in addition to the analysis of vibration measurements. Condition monitoring systems for in-depth diagnosis are based on wired acceleration sensors, which are connected with diagnosis core units. There are only few wireless system solutions to run the less precise base diagnosis of machines.

The concept in the joint project ECoMoS (see Fig. 1) is based on wireless condition monitoring using wireless sensor nodes [8]. The individual sensor nodes have the intelligence to carry out an in-depth diagnosis directly at the measuring position. This makes it possible to predict, for example, bearing damage by up to three months in advance. The sensor systems can be configured via radio transceiver and regularly send the machine state to a remote base station.

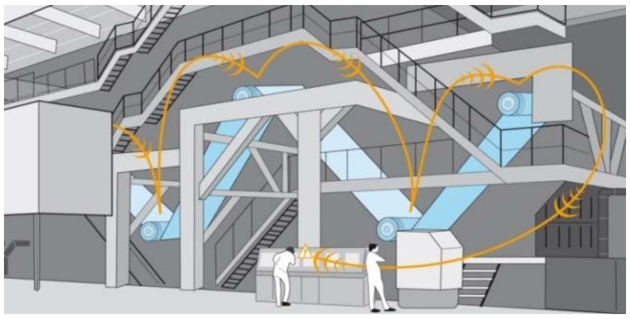


Figure 1. Condition monitoring of plants by wireless sensor networks

The complexity of the in-depth diagnosis drives the system requirements for the wireless sensor nodes to a large extent [9]. The vibration signal needs to be sampled at a frequency of 10 kHz and then digitized. At the same time, it is required to check whether the engine speed has remained within a tolerable range of about 0.5 percent during the measurement interval. If the change in rotational speed is greater, the measurement will be discarded. Since the measurement of the engine speed is performed by a different sensor, the speed values are transmitted before and after the vibration measurement of the wireless sensor node.

The spectrum of the measured time signal has to be calculated by a Fast Fourier transform (FFT) [10] to later derive the envelope spectrum of the vibration signal. Whenever the engine speed remains in a tolerable range, the order spectrum can be determined by replacing the frequency in the spectrum abscissa with the basic rotational speed. The critical multiples of the basic rotational speed are identified by a significance analysis. In a subsequent step, it is checked whether these coincide with typical damage patterns. If errors are detected, e.g., irregularity on the inner ring of the roller bearing SKF 6309, the results of the diagnosis are transmitted in the form of feature vectors, comprising error code, kinematic frequency, significance level, and magnitude.

For the implementation of wireless sensor nodes that can perform this type of machine diagnostic directly at the measuring position, different circuit designs were evaluated regarding their energy ficiency. Some Digital Signal Processors (DSP) were capable of performing the algorithms for in-depth diagnosis more energy-efficiently than 16-bit microcontrollers, e.g., MSP430 from Texas Instruments, or 32-bit microcontrollers, e.g., AVR32 from Atmel. Hence the power consumption of the processor types could be compared for a number of FFT cycles with 1024 samples at a clock frequency of 1 MHz (see Fig. 2). While the DSP based system architectures require significantly higher peak currents, the execution time is considerably shorter. This results from the special hardware support for filters and FFT calculations.

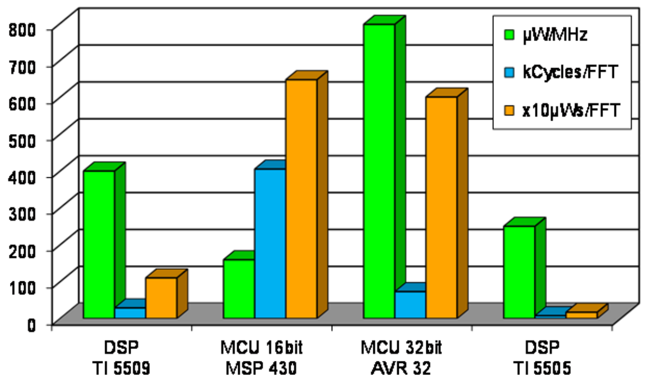


Figure 2. Comparison of different system architectures [11]

The analysis of analog circuit elements for data acquisition showed that the use of a DSP system architecture is the best solution to minimize the power consumption because the data from the acceleration sensor can be digitized immediately while the necessary filtering and scaling is left to the DSP. Another advantage of this approach is the increased flexibility in the evaluation of the measurement. It is thus possible to generate customized reports for specific measurement positions by parameterization with moderate effort. Fig. 3 summarizes the system architecture of the ECoMoS sensor nodes.

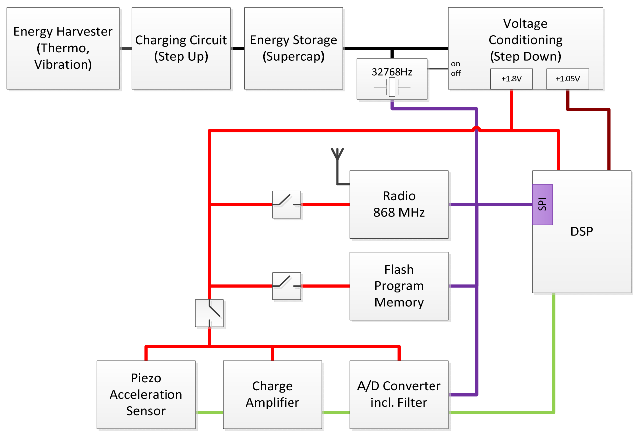

Energy Harvester (Thermo, Vibration)
Charging Circuit (Step Up)
Energy Storage (Supercap)
32768Hz
on
off
Voltage Conditioning (Step Down)
+1.8V
+1.05V
Radio 868 MHz
SPI
DSP
Flash Program Memory
Piezo Acceleration Sensor
Charge Amplifier
A/D Converter incl. Filter

Figure 3. System architecture of the radio sensor nodes [11]

The selection of the appropriate sensor depends on accuracy of the measurement range, size, supply voltage, and power consumption. It was found that a piezoelectric acceleration sensor from IMI Sensors with charge output provides the preferred properties for the target application. Thus, a charge amplifier has been added to the input stage on the sensor node. This makes it possible to sample the machine vibrations with minimal energy consumption. Furthermore, the duration of the transient can be considerably shortened by a special discharge stage. This would not be possible for sensors with integrated amplifier.

III. Electric Energy from Vibrations and Temperature Differences

Research and development in the area of micro-scale MEMS transducers is still ongoing with future potentials to satisfy steadily decreasing energy demands of microelectronic circuits [1]. However, depending on the ambient conditions available energy harvesting solutions with size constraints of a few centimeters already provide adequate voltage and current level. While solar cells achieve high power densities in direct sunlight, stability of solar irradiance is a challenging issue especially when designing technical solutions for a broader range of possible applications. Artificial lighting conditions, however, provide rather predictable illumination conditions, e.g., considering public buildings, workspaces, etc. Amorphous solar cells will deliver around 3µW/cm^2 at fluorescenting light with an illuminance of 200lux (machine hall) [12]. In case kinetic energy from vibrating objects such as engines, buildings or vehicles is available, resonant electromechanical transducers might be best suited. Piezoelectric transducers based on bimorph cantilever structures oscillating at resonant frequency of 120Hz and amplitude of vibration at 2.25m/s^2 deliver a power density of 375µW/cm^3 on an ohmic load

[13]. Macro-scale assemblies of thermoelectric harvesters with total dimensions in the range of cubic centimeters consist of leg pairs of doped semiconductors that are mounted between two ceramic substrates and electrically connected in series. Subjected to a temperature gradient of 5K at 50°C a power density in the order of 590µW/cm² can be achieved [14].

As all harvesting technologies deliver an unstable output voltage depending on the available profile of energy inputs, a voltage converter is necessary in order to adapt to the requirements of the consuming electronics. In order to buffer energy, ultra-capacitors, accumulators or even a supporting primary battery might be an option, depending on the envisioned application. The storage unit should be designed to enable operation at times when ambient energy is not sufficiently available, but also to allow duty-cycle operation of the system. Another voltage converter at the output terminals of the energy buffer will regulate the output voltage provided to the energy consuming subsystems. A categorized schematic of the energy harvesting conversion chain is shown in Fig. 4.

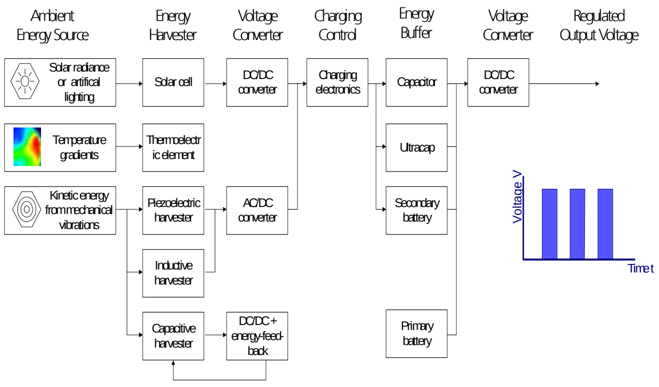


Figure 4. Schematic of the energy harvesting conversion chain

The introduction of energy harvesting technologies to wireless sensor nodes for condition monitoring purposes has the potential to overcome the current conflict between desired operating time and functionality [15]. To replace conventional battery-based solutions with energy harvestings systems on a broad basis, system behavior under realistic ambient conditions needs to be described by adequate models. Design approaches for self-sufficient sensor systems rely on the provision of sufficient power levels according to the specific boundary conditions determined by the application. In order to sufficiently describe the dynamic behavior of energy harvesting systems over its operating range, a modeling framework according to Fig. 5 was implemented as presented in [16]. Components of the conversion chain are modeled using parameter sets acquired from measurement to fully describe the devices over their operating range. The parameterized sub-models are then transformed into modeling platforms, e.g., Matlab/Simulink that provide capabilities for mutual modeling of different but connected physical domains. As a result, energy flows can be simulated based on the time history of the non-electric ambient energy source's intensity and the specific load profile of the power consuming electronics.

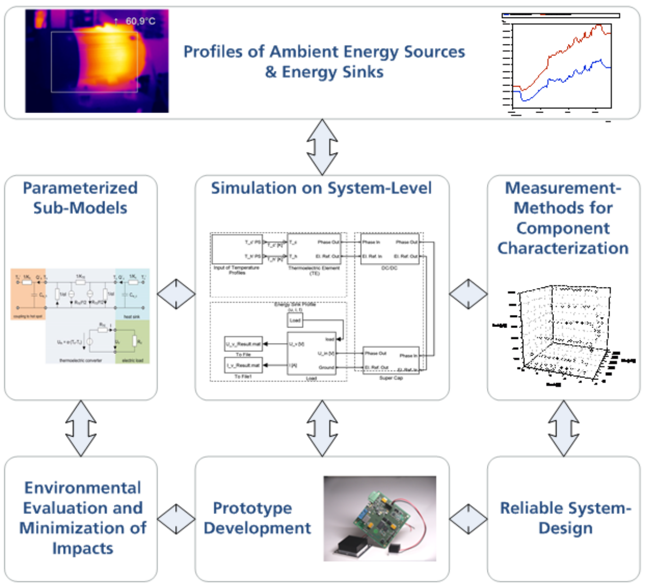

Profiles of Ambient Energy Sources & Energy Sinks
Parameterized Sub-Models
Simulation on System-Level
Measurement-Methods for Component Characterization
Environmental Evaluation and Minimization of Impacts
Prototype Development
Reliable System-Design

Figure 5. Concept of system-oriented modeling approach

The operating time of the wireless sensor nodes with batteries – depending on the duty cycle of the in-depth diagnoses – can reach several years by application of low-power components in combination with an optimized power management [8]. Cycles of sleep-mode vs. active phase need to be adapted to the available power budgets. Fig. 6 exemplarily demonstrates the voltage level monitored at an ultra- capacitor used as an energy buffer in the application case described in this paper.

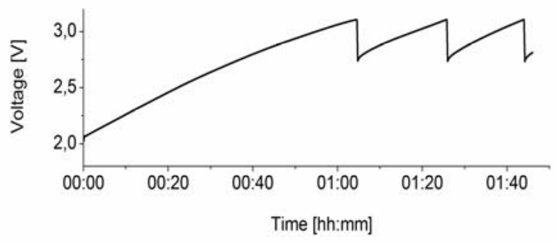


Figure 6. Voltage profile of ultra-capacitor during duty-cycle operation of the sensor node; field measurement

However, there are several concerns regarding battery-powered wireless sensors especially in industrial environ-ments. The users prefer energy harvesting solutions when-ever possible. While solar cells can only be used in very clean industrial surroundings with little dust, thermal gene-rators and vibration transducers provide a good way to convert the energy from the environment. This eliminates any effort for battery replacement and increases the operational reliability. As thermal gradients and mechanical vibrations provide useful energy sources in industrial environments, concepts for a thermoelectric and piezo-electric energy harvesting device are described exemplarily in the following sections.

There are many different types of vibrations that can be used for powering wireless sensor nodes in industrial environments. However, rigid structures should be chosen as measurement position for in-depth diagnosis of machines. This improves the precision of the diagnosis results but the magnitude of vibrations on these locations is typically below 2m/s² (see Fig. 7). Most commercial vibration transducers are designed for a single resonance frequency of between 50 and 120 Hz. This means that such a narrowband converter cannot be used on each machine, since the efficiency massively decreases whenever the vibrations are not in the vicinity of the resonance frequency.

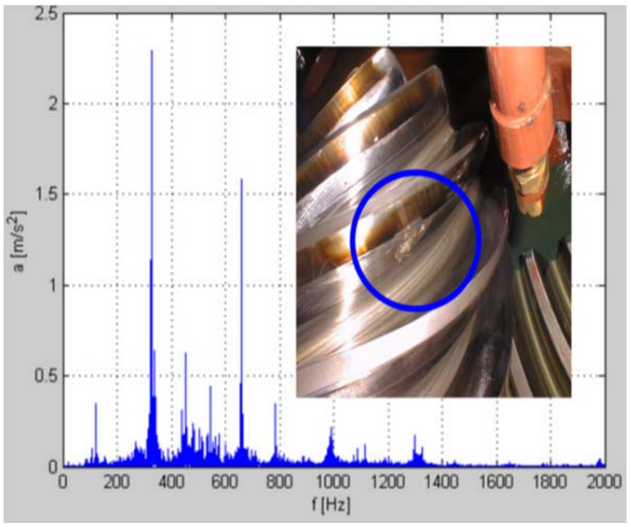


Figure 7. Exemplary acceleration spectrum of a typical drive

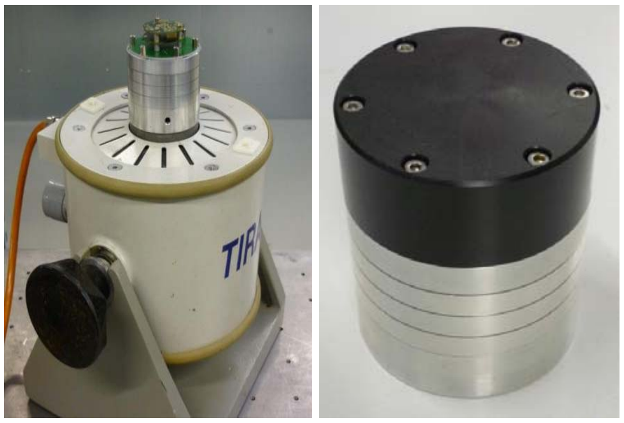


Figure 8. Prototype implementation of the vibration harvester, source: Baumer

A broadband vibration transducer (see Fig. 8) has been developed in the ECoMoS project by skillful stacking of piezo elements. The first prototypes of the vibration energy converter (see Fig. 9) were tested with a tunable Shaker. It was found that the vibrations in the range of 300 to 800 Hz provided enough energy to supply the radio sensor electronics. Such broadband transducers can be used more universally.

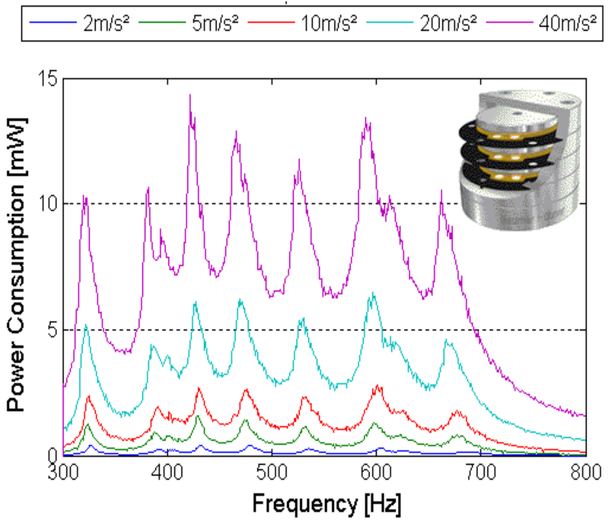
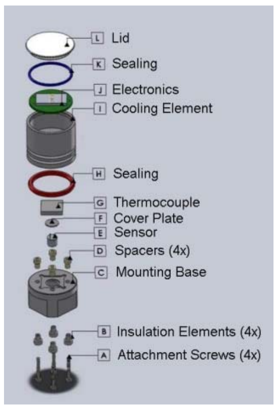


Figure 9. Design of a broadband vibration transducer by stacking piezo elements [10]

Significant temperature differences can often be found in industrial plants. Different drives were examined under operational conditions regarding their temperature distribution to develop very compact wireless sensor nodes with an integrated thermal converter. The temperature differences that occur directly on the thermocouple were observed for typical measuring points using a special test setup consisting of a reference heat sink and thermal converters with embedded temperature sensors. A temperature difference of only 3°C at the thermocouple was required to deliver enough energy for the power consumption of wireless sensor nodes for condition monitoring.

The developed concept for the sensor node shown in Fig. 10 provides integrated thermoelectric energy conversion capabilities allowing for fully autonomous operation. The concept is based on dual usage of the housing as both, protective holding of sensitive electronic components and the thermal path. In the center, a Bi_2Te_3-based thermoelectric device with edge-length of 20mm is mounted in between a bottom mounting base and a top cooler with integrated cavity for inclusion of the electronic system. The aluminum body with 58mm in diameter and a total height of 45mm is then mounted to a hot surface allowing for direction of heat flow orthogonal to the plane coupled to the hot spot. Screws for fixture of cooler and mounting base include thermally isolating sleeves and an isolating spacer defining a fixed height for thermoelectric module inclusion. Silicone-based heat-conductive pads are included on hot and cold sides of the thermoelectric modules to account for height tolerances by the manufacturer and provide damping when opposing the module to mechanical loads.

a) 

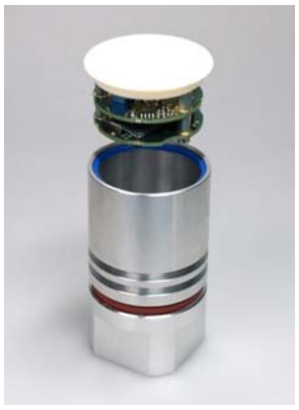
b) 

Figure 10. a) Assembly concept of the radio sensor system with integrated thermal energy havesting, b) Implemented prototype

Energy harvesting devices combine properties of different physical domains. The development of a multiphysics model suited to describe these domains simultaneously is therefore recommended. For the thermoelectric converter a suited model comprises the thermal path through couplings to heat source and –sink as well as the electrical side. Thermal voltage generated

$$U_{th} = \alpha_S \cdot (T_h - T_c) = \alpha_S \cdot dT \qquad (1)$$

depends on the Seebeck coefficient α_S and the thermal difference between hot- and cold-side temperatures T_h and T_c respectively across the thermoelectric element. However, the thermoelectric figure of merit

$$ZT = \frac{\alpha_S^{\;2}}{\lambda \cdot \rho} T \qquad (2)$$

is a measure of a material's capability to efficiently generate electric power from temperature gradients. It depends on the thermal conductivity λ, and electrical conductivity ρ and absolute Temperature T.

By transformation using the analogy between thermal and electrical path a mutual modal can be derived that enables computation of electric voltage and current depending on the heat flow through the device as well as the peripheral electric circuitry. The setup of the electronic subsystem for voltage conversion and buffering was included in the analysis to simulate performance on system level.

Temperature distribution at the measuring points was applied in simulations to evaluate different assembly concepts (see Fig 11). In order to optimize the thermal properties of the housing computational fluid dynamics (CFD) software assuming varying ambient conditions was applied. The model was then used to calculate available temperature levels within the construction considering thermal interfaces and coupling to hot and cold sides. For this purpose a geometry model of the proposed solution was set up in ANSYS Icepak with the thermoelectric device being modeled with the elements shown in [10]. Material properties of metals and isolators were taken from Icepak library while specific heat conductivity of the thermoelectric element as well as the thermal interface material were determined from experiment. Boundary conditions resembling possible field conditions were set after applying a standard hexahedral mesh to the model.

All modules were aligned orthogonally to the z plane with the main direction of heat transfer in the z-direction. A constant temperature of T=61°C resembling available hot surfaces of industrial type engines was applied to the lower x-y-plane of the cabinet. The ambient temperature of surrounding medium at initial state was set to 35°C as measured in the machine hall. For simulation of natural convection effects, activation of gravity in the z-direction was considered. In the x-direction, the flow of fluid (air) particles was increased in four steps. In the worst case, the sensor node has to function at a solely natural convection. A typical velocity of v=0.2m/s was then increased in discrete steps to values of 0.5m/s and 1m/s resembling forced convection in the immediate environment of the devices, e.g., fans in the motor block available within immediate distance from the mounting base.

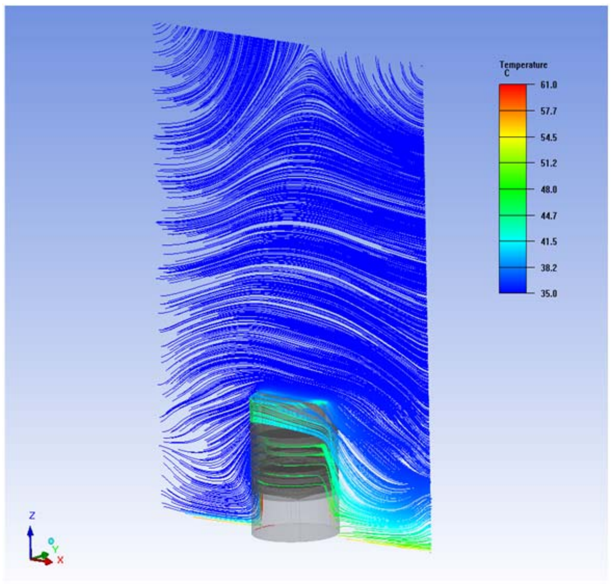


Figure 11. Simulation result of temperature distribution for a thermo-electric harvesting device at forced convection of v=1m/s

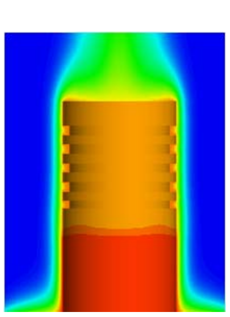
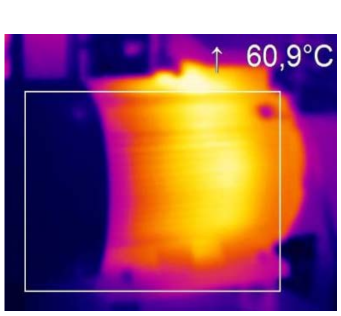
a) 
b) 

Figure 12. a) Simulation of temperature distribution, b) Temperature distribution during machine operations

Temperature distribution of running engines was determined by infrared-thermography (see Fig. 12). A robust housing with a diameter of only 4 centimeters was designed so that the hot and cold sides of the thermal converter are coupled particularly well with the industrial environment, without reducing the accuracy of the high-precision acceleration sensor.

Fig. 13 demonstrates the exemplary results of a simulation applied to the demonstrator. Average power provided to the consuming electronics at a constant voltage of 2.1V was calculated based on the simulation results of the system-oriented modeling concept using Matlab/Simulink (see Fig. 5).

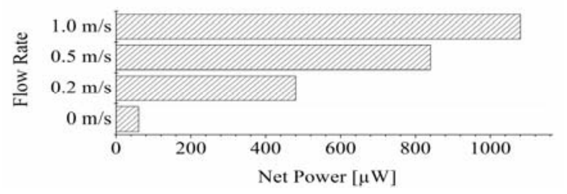
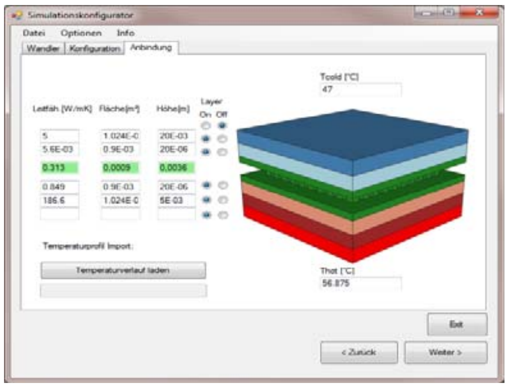


Figure 13. Continuous net power generated by the thermoelectric energy harvester

In the case of natural convection, only copper-based housing provided sufficient heat transfer within the thermal path. Choosing this option, voltage converted at the thermoelectric device was just above threshold value to start up the power management IC. Efficiency of the demonstrator was verified using a wind tunnel test setup with deviations below 10% from energy output simulation results.

A design tool for a first order approximation has been implemented to facilitate the future development of wireless sensor systems with embedded energy harvesting. This software provides the system specific parameters from the component library. The parameters can be adjusted by the hardware designer to layout the power supply of self-powered wireless sensor nodes. After selection of the conversion method, the type of physical size of the harvesting unit can be suitably configured. In the example of the thermal converter, this applies to additional layers, (e.g., thermal paste and mounting adapter), which have to be taken into account for their impact on the energy efficiency (see Fig. 14).



Tcold [°C]: 47

Leitfäh [W/mK]	Fläche[m²]	Höhe[m]	Layer On	Layer Off
5	1.024E-0	20E-03		●
5.6E-03	0.9E-03	20E-06	●	
0.313	0.0009	0.0036		
0.849	0.9E-03	20E-06	●	
186.6	1.024E-0	5E-03	●	
			●	

Thot [°C]: 56.875

Figure 14. Selection of the energy converter and configuration

Following the energy conversion chain, the parameters for the voltage conditioning and energy storage can be queried. It is possible to select predefined voltage transformers. During simulation the input or output impedance at the operation point as well as the converted output voltage or current level are considered together with non-ideal effects of energy buffers such as leakage current and a parasitic internal resistance. In the input mask for the configuration of the load, the clock frequency of the processor is to be set in order to determine the computation time for the individual algorithms. This approach allows for a first estimation of the energy performance of the wireless sensor node to determine the size of the power conversion stage and energy buffer so that particularly compact sizes can be realized inexpensively. The method can be seen as a supplementary tool to detailed system analysis as previously described. It is also planned to use System-C for system modeling to investigate more complex algorithms in terms of energy requirements.

IV. Installation and Maintenance per Wireless Network

In production halls, there are many sources of interference that can affect radio transmission considerably. In the early days of radio communication, many problems regarding robustness of radio links occur in the industrial sector. This led initially to a high skepticism about wireless sensors, because the robustness of data transmission in industrial environments is particularly important. Meanwhile, there are various solutions to ensure robust radio communications. In particular, procedures at the protocol level of digital radio communication can reconstruct the correct data even when data erasure of multiple bits occurs. Therefore, the problem of robust data transmission is well under control in this manner, although the increase in energy efficiency of robust radio communication still remains an important research topic. An energy-efficient medium access control protocol based on Time Division Multiple Access [17] was applied here. In this single-hop network, all sensor nodes receive a time schedule about the next communication interval from the base station.

Security and trust are very important issues in industrial environments. The corresponding measures are currently under development. A secure wireless communication among the sensor nodes and the gateway will be ensured by AES encryption and unique identifiers for each device. The communication between the gateway and the private cloud server will be implemented using the OPC UA protocol stack. To assure a secure data access of the different users, a commercial solution such as CodeMeter® will be chosen for the license management in the cloud. This security approach guarantees diagnostic results based on trusted sensor data.

The installation of wireless sensors can now be done quite comfortably. After attaching to a machine, sensor nodes identify existing wireless networks and log on. The operating parameters of the sensor systems, such as duty cycles, can be adjusted via the radio base station. A software update via radio link is also possible if new damage patterns or changes in communication protocols require reprograming the wireless sensor nodes. Thus, the user does not need to trigger a time-consuming firmware update for each individual sensor nodes via cable. Such a wired reprograming of sensor node can be very expensive especially for measurement positions, which are hard to reach.

For the preparation of the field tests, the prototypes were first tested in the laboratory by a tunable Shaker, which simulated the vibration profile of broken machines. The direct field trials were carried out in the harsh environment of a paper mill under realistic operating conditions in order to obtain more meaningful test results.

On the site of the former gas conglomerate, Schwarze Pumpe, between Cottbus and Dresden, where earlier lignite was refined, paper production has been in full swing since April 2005. 170 million euros were invested to build the Spremberg mill. With a capacity of 330,000 tons of base paper per year, corrugated board is produced from 100 percent recycled paper. The preparation of a 1000 meter long and 5.4 meter wide paper web requires just a minute. Only one hour of downtime would cost 5,000 euros. Wireless technology significantly reduces material and installation costs for copper wire and cable channels for about 100 meters total length of the paper machine (see Fig. 15), that the initial investment for a wireless condition monitoring system offers a very short payback time.

Figure 15. Paper plant in Spremberg [10]

The fusion of data from the wireless sensor nodes is provided by the base station, the so-called ECoMoS server, which is connected by field bus with the control center of the paper machine (see Fig. 16). During the installation of sensor node as well as the exchange of production components the kinematic parameters for the in-depth diagnosis has to be updated at the ECoMoS server. The ECoMoS server also receives the rotation speeds of the individual drives in the paper machine and transmits this data to the corresponding sensor nodes ted to the server ECoMoS. This procedure is done before and after each measuring interval.

If a damage pattern is detected, each sensor node can send the raw data via the base station and the ECoMoS server to the control center of the paper plant. So, experts can examine the characteristics of serious errors in critical cases before costly actions for damage repair are initiated. Often, it is sufficient to replace a damaged bearing during the regular maintenance interval.

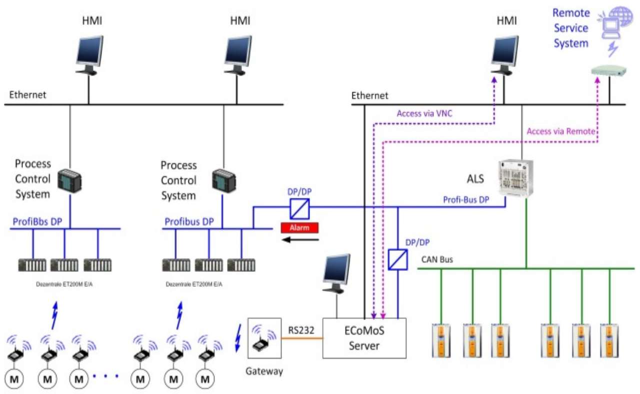


Figure 16. Integration of the ECoMoS sensor network into the process control system of the paper plant [10]

V. SUCCESSFUL PRACTICAL TEST IN THE PAPER FACTORY

In the first prototype generation, the system parts sensor, data processing, radio interface and power supply had been designed as modules to be tested in the paper mill separately. The final prototype generation as a compact complete system was produced in February 2014. Currently, the last work on the wireless protocol and the software for the ECoMoS server ECoMoS is nearly finished. Firstly, four measuring positions in the paper mills were equipped with the ECoMoS wireless sensor nodes to demonstrate a successful project completion (see Fig. 17). This sensor network can be extended to the full configuration with 664 sensor nodes.

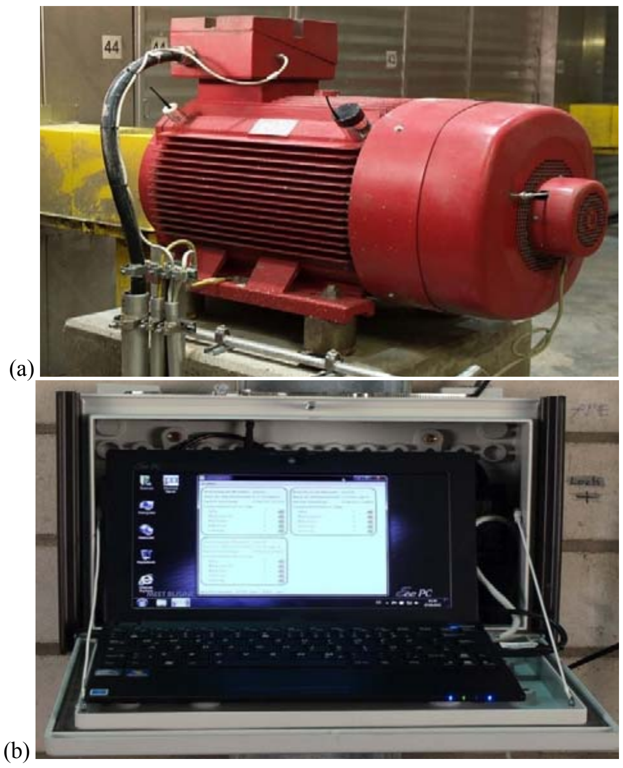


Figure 17. a) Field tests of the second prototype generation, b) ECoMoS-Server in operation

The wireless sensor network, developed by the ECoMoS project, can be used to detect wear of machines and plants that have drives, which are at least temporarily operated at a constant speed. There are many such manufacturing facilities. When in the future, low-cost wireless sensor systems will be available, condition monitoring will be used for additional rotating parts such as fans, which are not currently monitored due to cost reasons. The bill of material for such sensor node ranges from 10 to 300 euros depending on the requirements and environment. The cost of an entire sensor network, however, is mainly influenced by the production quantity. The development costs dominate the overall cost in niche applications. For a wide range of applications, however, such wireless sensor nodes will be produced in large quantities, so that the design and software costs only account for a small proportion of total costs.

VI. Conclusion and Future Work

The aim of the joint project ECoMoS was to develop a wireless sensor network for the condition monitoring of industrial machines. The wide range of expertise in the fields of measurement technology, wireless communication, energy harvesting, module integration and machinery diagnosis were necessary to demonstrate the opportunities of modern wireless sensors for condition monitoring. The installation of sensor networks within plants provides the basis for advanced concepts in condition monitoring. Wireless solutions for machine diagnosis make it possible to reduce installation costs while collecting and evaluating more data on the individual sensors.

As such systems become available, the leverage effect for the construction of machines and plants becomes hard to predict. Such sensor networks will be connected with cloud services, so that the machine diagnosis can be supported by external service providers or by the system manufacturer. The manufacturer would then be in a much better position to assess the reliability of their products. It can be observed, for instance, whether or not the construction of machine components needs to be improved, or whether it is possible to reduce costs, without impairing the reliability.

Even if the use of radio sensor systems with artificial intelligence in production facilities is still to be regarded as visionary, highly miniaturized wireless sensor nodes for the future self-organization of production systems are under development.

Acknowledgment

The joint project ECoMoS was funded by the Federal Ministry of Education and Research (BMBF) and managedwith support of VDI / VDE Innovation + Technik GmbH in Berlin. The project results has achieved by the cooperation of the companies IMC Meßgeräte, Elbau, GfM Gesellschaft für Maschinendiagnose, GE Energy, Baumer, EnOcean, Hamburger Rieger together with the Fraunhofer IZM and the Technical University Berlin.

References

[1] M. Niedermayer, E. Kravcenko, R. Wirth, A. Haubold, S. Benecke, and K. D. Lang, “Early-Warning System for Machine Failures: Self-Sufficient Radio Sensor Systems for Wireless Condition Monitoring,” Proc. of 7th Int. Conf. on Sensor Technologies and Applications SENSORCOMM, Barcelona, Spain, pp 225–230.

[2] M. Niedermayer, R. Thomasius, D. Polityko, K. Schrank, S. Guttowski, W. John, and H. Reichl, “Miniaturization Platform for Wireless Sensor Nodes Based on 3D-Packaging Technologies,” Proc. of IEEE Int. Conf. IPSN/SPOTS, Nashville, TN, Apr. 2006, pp 391–398.

[3] M. Niedermayer, J. Hefer, and S. Guttowski, “Technology-Oriented Design of Radio Sensor Nodes,” in Smart Systems Integration and Reliability, Ed. B. Michel, Goldenbogen, ISBN: 3540238670, Dresden, Germany, 2010.

[4] M. Niedermayer, C. Richter, J. Hefer, S. Guttowski, and H. Reichl, “SENESCOPE: A Design Tool for Cost Optimization of Wireless Sensor Nodes,” Proc. of IEEE Int. Conf. IPSN/SPOTS, San Francisco, CA, Apr. 2009, pp. 313–324.

[5] M. Niedermayer, “Commercialisation and Application Driven Economic Viability of Sensor Technology,” in Comprehensive Materials Processing, Ed. S. Hashmi, Elsevier Science Ltd, ISBN: 0080965326, Oxford, UK, 2013.

[6] E. Hohwieler, E. Uhlmann, M. Niedermayer, and B. Hirsch, “Self-organizing Production with Distributed Intelligence,” Proc. 2nd Int. Symp. on Innovative Production Machines and Systems, Ischia, Italy, May 2009, pp. 163–167.

[7] M. Niedermayer, H. Scholtz, T. Bonim, S. Guttowski, and K. D. Lang, “Robust 3D Radio Sensor Systems with Embedded Active and Passive Components for Industrial Applications,” Proc. Int. Conf. on Sensors and Its Applications SIA, Jeju Island, Korea, Nov. 2012, pp. 336–341.

[8] E. Kravcenko, M. Niedermayer, S. Guttowski, N. Nissen, S. Benecke, A. Middendorf, and H. Reichl, “A System-oriented Approach for Modeling Energy Harvesting Devices in Wireless Sensor-Modules, ” Proc. of 4th Int. Conf. on Sensor Technologies and Applications SENSORCOMM, Venice, Italy, Jul. 2010, pp. 305–310.

[9] M. Niedermayer, “Cost-Driven Design of Smart Micro Systems,” Artech House Publishers, ISBN: 1608070840, Norwood, MA, U.S. 2012

[10] E. O. Brigham, “The Fast Fourier Transform,” Prentice-Hall, New York, 2002.

[11] R. Wirth, “Self-Sufficient Condition Monitoring System ECoMoS,” Grant 16SV3371-16SV3377, Project Report, TIB Hanover, 2013.

[12] Solar Cells Technical Handbook ‘98/99, Panasonic, Industrial Co., Ltd.

[13] S. Roundy, P. K. Wright, J. Rabaey, “Energy Scavenging for Wireless Sensor Networks with Special Focus on Vibrations,” Kluwer Academic Publishers, 2004.

[14] Product information TEG-127-150-26, www.thermalforce.de, March 2010.

[15] S. Sudevalayam, and Kulkarni, P., “Energy Harvesting Sensor Nodes: Survey and Implications. IEEE Communications Surveys and Tutorials,” Vol. 13, No. 3, 2011, pp. 443–461.

[16] S. Benecke, J. Rueckschloss, A. Middendorf, N.F. Nissen, and K.-D. Lang, “Energy Harvesting for distributed microsystems – The link between environmental performance and availability of energy supply,” Proc. EcoDesign, 2011.

[17] S. C. Ergen and P. Varaiya, “TDMA scheduling algorithms for wireless sensor networks,” Springer Wireless Network, Vol. 16, 2010, pp. 985–997.

Low Energy Adaptive Clustering in Wireless Sensor Network Based on Spectral Clustering

Ali Jorio*, Sanaa El Fkihi †, Brahim Elbhiri‡, and Driss Aboutajdine*
*LRIT, Research Unit Associated to the CNRST (URAC 29), FSR, Mohammed V University, Rabat, Morocco
jorio.ali@gmail.com, aboutaj@fsr.ac.ma
†ENSIAS Mohammed V University, Rabat, Morocco
elfkihi@ensias.ma
‡EMSI Rabat, Morocco
elbhirij@emsi.ac.ma

***Abstract*— Wireless Sensor Networks (WSNs) are composed of large number of sensor nodes that are randomly distributed in a region of interest. The nodes are responsible of the supervision of a physical phenomenon and periodic transmission of results to the sink. Energy saving results in extending the life of the network, which presents a great challenge of WSNs. This paper focuses primordially on reducing the power consumption of WSN. To deal with this, a hierarchical clustering scheme, called Multi-Relay K-way Spectral Clustering Algorithm (MR-KSCA), is proposed. This algorithm is based on spectral classification and graph theory with the aim to cluster the network in a fixed optimal number of clusters. Thus, our approach ensures an ideal distribution of sensor nodes in clusters, and proposes new features to elect the appropriate cluster-heads, which guarantee an uniform distribution load of energy among all the sensor nodes. Furthermore, We present three metrics to define the WSN lifetime. In term of extending the network lifetime and minimizing the energy consumption, the simulation results show an important improvement on the network performances with MR-KSCA compared to other existing clustering methods.**

Keywords-Wireless Sensor Network; Clustering; Graph theory; Spectral classification; Energy consumption.

I. Introduction

A Wireless Sensor Network (WSN) is a wireless network consisting of spatially distributed autonomous devices using sensors to cooperatively monitor physical or environmental conditions at different locations, such as temperature, sound, and vibration. It contains a large number of nodes, which sense data from an inaccessible area and send their reports towards a processing center called Base Station (BS) or Sink. WSNs have many advantages, such as the ease deployment and the capacity of self-organization. However, their main challenges are the limited resources in terms of radio communication capabilities and energies. This article addresses this last point. it is based on [1] that was presented in SENSORCOMM 2013.

WSN has been widely used in many areas especially for surveillance and monitoring in agriculture and habitat monitoring. Environment monitoring has become an important field of control and protection, providing real time system and control communication with the physical world [2]. In this paper, we propose an energy efficient cluster based routing algorithm for environmental monitoring system.

Even if the sensors are usually powered by batteries, it is not practical to recharge or replace them, because they are often deployed in hostile environments. Furthermore, in a WSN, a large part of energy is consumed when communications are established [3]. Hence, frequent and long distance transmissions should be minimized to extend the lifetime of the network. One way to reach this result consists in dividing the network into multiple clusters; each cluster has a cluster-head (C-H) [4]. The C-H node collects data from nodes of its same cluster, aggregates them and transmits them to the BS. Nevertheless, the main problem of many proposed clustering protocols is the random selection of the C-H nodes [5]. Indeed, all C-Hs can be located in a small region of the network, and some ordinary nodes will be isolated. Also, the nature of the clustering problem is such that the ideal approach is equivalent to finding the global solution of a non linear optimization problem. This is usually a difficult task to achieve. In actual fact, this is an NP-hard problem [6]. To tackle the clustering problem and to fairly localize C-H nodes, we propose to consider spectral clustering methods and the sensor's residual energies.

Spectral clustering has become one of the most popular modern clustering algorithms. It is based on graph theory, can be solved efficiently by standard linear algebra software. It also outperforms traditional clustering algorithms such as the k-means algorithm [7]. Spectral clustering is refereed to a class of techniques, which rely on the eigen-structure of a similarity matrix to partition points into disjoint clusters (i.e., points in the same clusters have high similarity and points in different clusters have low similarity [8]. These spectral clustering algorithms have attracted attention over the past few years in many applications, such as image segmentation [9] and social network analysis [10]. The K-way approach is one of these methods. It consists to divide points into K disjoint classes. This method is based on the K eigenvectors related to the K largest eigenvalues of Laplacian matrix [11].

Following this approach, we present a Multi-Relay K-way Spectral Clustering Algorithm (MR-KSCA) in Wireless Sensor Networks. MR-KSCA consists in partitioning the Wireless Sensor Network into a set of clusters based on spectral classification. Besides, it determines the C-H nodes and super cluster-head node (super C-H) based on the residual energy of each node of the network. Some other properties will be explained in Section IV.

Thus, MR-KSCA has four main goals: (i) extending the network lifetime by distributing energy consumption, (ii) running the clustering process within a constant number of iterations/steps, (iii) minimizing control overhead (to be linear in the number of nodes), and (iv) producing well-distributed C-Hs and compact clusters. MR-KSCA does not make any assumptions about the distribution or density of nodes, neither about node capabilities.

We test the proposed algorithm through simulations and compare the results with other clustering algorithms. Results show that our proposed method is far better than the others With respect to energy consumption and network lifetime.

The remaining of this paper is organized as follows: In Section II, we give a brief overview of some related research work. Section III describes the network model and states the addressed problem in this work. Details and properties of the proposed algorithm are given in Section IV while Section V presents the simulations and the results. Finally, conclusion and some perspectives are drawn in Section VI.

II. RELATED WORK

Many routing protocols have been designed based on clustering [5][12][13], While there are advantages to use distributed cluster formation algorithms, these protocols offers no guarantee about the placement and the number of the C-H nodes. However, using a central control algorithm to form the clusters may produce better clusters by dispersing the C-H nodes throughout the network. This is the basis of our proposed algorithm.

Moreover, the most commonly used approach for clustering is the Low-Energy Adaptive Clustering Hierarchy (LEACH) algorithm [14]. LEACH is an energy-efficient communication protocol, which employs a hierarchical clustering. Besides, many clustering protocols based on the principle of this algorithm have been developed, such as LEACH-Centralized (LEACH) [15], Energy-Kmeans [16], DECSA [17], EECS [18] and SECA [19].

In LEACH, nodes organize themselves into clusters using a distributed algorithm. Its main idea is to randomly and periodically select the C-H nodes. The probability of becoming a C-H for each round is chosen to ensure that every node becomes a C-H at least once within N/K rounds, where N is the number of nodes in the network and K is the desired number of clusters. Thus, In order to select C-H nodes, each node i determines a random number between 0 and 1 if the number is less than threshold $T(i)$, the node becomes a C-H for the current round. The threshold is set as follows:

$$T(i) = \begin{cases} \frac{K}{N-K*\left(r*mod\left(\frac{N}{K}\right)\right)} & : \ C(i)=1 \\ 0 & : \ C(i)=0 \end{cases} \tag{1}$$

$C(i)$ is the indicator function determining whether or not node i has been a C-H in the most recent $\left(r * mod\left(\frac{N}{K}\right)\right)$ rounds (i.e., $C(i) = 1$ if node i has been C-H and zero otherwise). After the election of C-H nodes, each ordinary node in the wireless sensor network determines its cluster by choosing the C-H that requires the minimum communication energy, based on the received signal strength of the advertisement from each C-H. Each ordinary node will choose to join the C-H which has the highest signal quality. Once the clusters are formed, each C-H node creates Time Division Multiple Access (TDMA) schedule. The C-H collects and aggregates information from sensors in its own cluster and passes on information to the BS. Although LEACH distributed clustering algorithm has advantages, using a central control algorithm to form clusters may produce better clusters by dispersing the C-H nodes throughout the network [14].

In LEACH-Centralized (LEACH-C), also described in [15], uses a centralized algorithm to form clusters. The BS determines the C-H nodes by computing the average node energy. The node with energy below this average can not be C-Hs for the current considered round. This ensures that the energy consumption will be effectively distributed among all the nodes.

Yong and Pei [17] present a Distance-Energy Cluster Structure Algorithm (DECSA) based on the classic clustering algorithm LEACH. This proposed approach considers both the distance between the nodes, the position of the base and residual energy of nodes. Its main idea is to partition the network into three levels of hierarchy to reduce the energy consumption of the C-H nodes. This results from the non-uniform distribution of nodes in the network and thus avoid direct communications between BS and C-H node that has minimal energy and is far away from the BS.

Elbhiri et al. [20] propose a new approach called the Spectral Classification based on Near Optimal Clustering in Wireless Sensor Networks. This approach is based on spectral bi-section for partitioning a sensor network into two clusters. Spectral bi-section method is based on second eigenvalue λ_2 of the Laplacian matrix of the graph to be partitioned. Median value of corresponding eigenvector of λ_2 is used to bi-part the graph [11]. The authors apply this method recursively to obtain the desired number of clusters. Then, for each cluster, a C-H is also elected. This task is assigned by rotating way between nodes without considering the residual energy of the nodes. Hence, this method cannot partition the network into any desired number of clusters and do not ensure an equitable distribution of energy consumption between sensor nodes.

III. RADIO MODEL AND PROBLEM STATEMENT

This paper discusses the periodical data gathering application [21], for which LEACH is proposed. Thus, the proposed clustering algorithms usually produce clusters of the same size. The C-H node consumes the same amount of energy during intra-cluster communications. Moreover, C-H election method based essentially on residual energy can obtain better energy efficiency than the method in which cluster heads are elected in turns or by probabilities. For this, we use the spectral classification that is widely used to solve graph partitioning problems. The spectral methods get their name from the spectral theories of linear algebra. This theories allows affirming the diagonalization of real symmetric matrices. It also justifies the decomposition of real symmetric matrices into eigenvalues in an orthonormal set of eigenvectors. Besides, the graph partitioning problem can be reduced to the resolution of a numerical system $Mx = \lambda x$. Solving this numerical system consists in seeking an orthogonal base of eigenvectors of the matrix M [11].

In our approach, the BS constructs the graph corresponding to the WSN based on the spectral clustering principle. Let x be an observation vector composed of the sensor network nodes. This vector can be represented by an undirected graph $G(V, E)$; where V is the set of vertices (sensor nodes) identified by an index $i \in \{1, \cdots, N\}$ (let N be the number of nodes) and E is the set of edges that link each two vertices (communication link). Let $A \in \Re^{N \times N}$ be the similarity matrix of the graph G. Each value of A is associated to each pair of the graph nodes (i, j). The total weight of incident edges to node i is given by $d_{ii} = \sum_{j=1}^{j=N} a_{ij}$. The degree matrix $D \in \Re^{N \times N}$ of G is a diagonal matrix defined by $D = [d_{ij}]$. Finally, the Laplacian matrix of the graph, as illustrated by [11], is expressed as follows :

$$L = D^{-\frac{1}{2}} * A * D^{-\frac{1}{2}} \tag{2}$$

To simplify the network model, we make some assumptions that are:

- All nodes are homogeneous and have the same capacities.
- Each node is assigned a unique identifier (id) that includes the cluster identifier to which the node belongs.
- The network topology remains unchanged over time.
- The BS is placed far away from the network.

In addition, we use the model of the radio hardware energy dissipation presented in [14][15], as illustrated in Figure 1. In this model, the multipath fading channel is used when the distance between the sender and the receiver is more than threshold d_0; otherwise the free space model is used. Therefore, in order to transmit an $L - bits$ message with

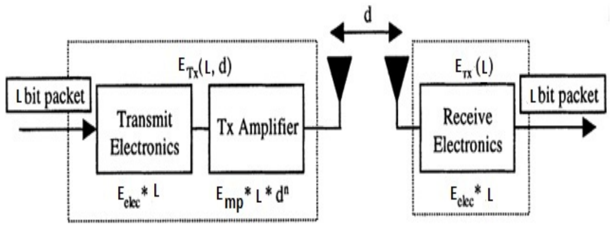


Fig. 1. Radio energy dissipation model.

distance d there would be:

$$E_{Tx}(L, d) = \begin{cases} L * E_{elec} + L * E_{fs} * d^2 & if\ d < d_0 \\ L * E_{elec} + L * E_{mp} * d^4 & if\ d \geq d_0 \end{cases} \tag{3}$$

To receive this message, the required amount of energy would be:

$$E_{Rx}(L, d) = L * E_{elec} \tag{4}$$

where E_{elec} is the energy consumption per bit in the transmitter and receiver circuitry, E_{fs} or E_{mp} is the energy dissipated per bit to run the transmit amplifier. The threshold's value (d_0) is expressed as follows:

$$d_0 = \sqrt{\frac{E_{fs}}{E_{mp}}} \tag{5}$$

IV. MR-KSCA Details

In this section, we give details of the proposed MR-KSCA algorithm (Figure 2). MR-KSCA considers the following clustering model: The sensor nodes are divided into several clusters by using spectral clustering method. For each resulting cluster, one sensor node is selected as a cluster-head. Then, from this set of C-H nodes, one node is selected as super C-H node. This super C-H node plays the role of virtual BS. Thus, the C-H nodes collect data from other sensor nodes in the cluster and transfer the aggregated data to the super C-H node or the BS, and the super C-H node collect data from the C-H nodes and transfer the aggregated data to the BS. Therefore, the data which are collected by the cluster head far away from the BS must be forwarded by the other C-H nodes that are close to the BS. Since data transfers to the BS dissipate much energy, the nodes take turns with the transmission: the role of cluster-head and super cluster-head "rotate". This rotation leads to a balanced energy consumption of all nodes and hence to a longer lifetime of the network. Some other properties and details will be explained in the subsections.

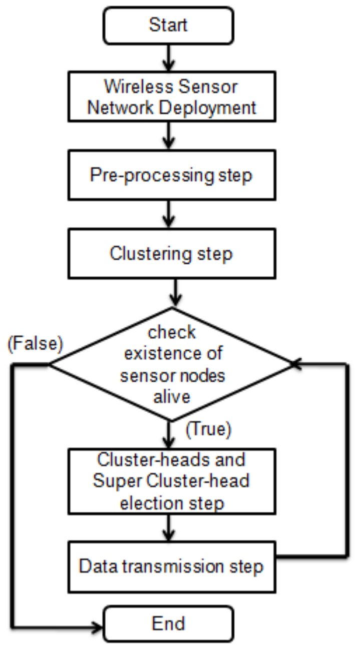

Start
Wireless Sensor Network Deployment
Pre-processing step
Clustering step
check existence of sensor nodes alive
(False)
(True)
Cluster-heads and Super Cluster-head election step
Data transmission step
End

Fig. 2. Flowchart of the proposed algorithm.

The spectral classification algorithm proposed here consists of three steps, as illustrated in Figure 3.

A. The Pre-Processing step

First, each sensor node determines its position by using a number of different technologies, e.g. exploiting radio or

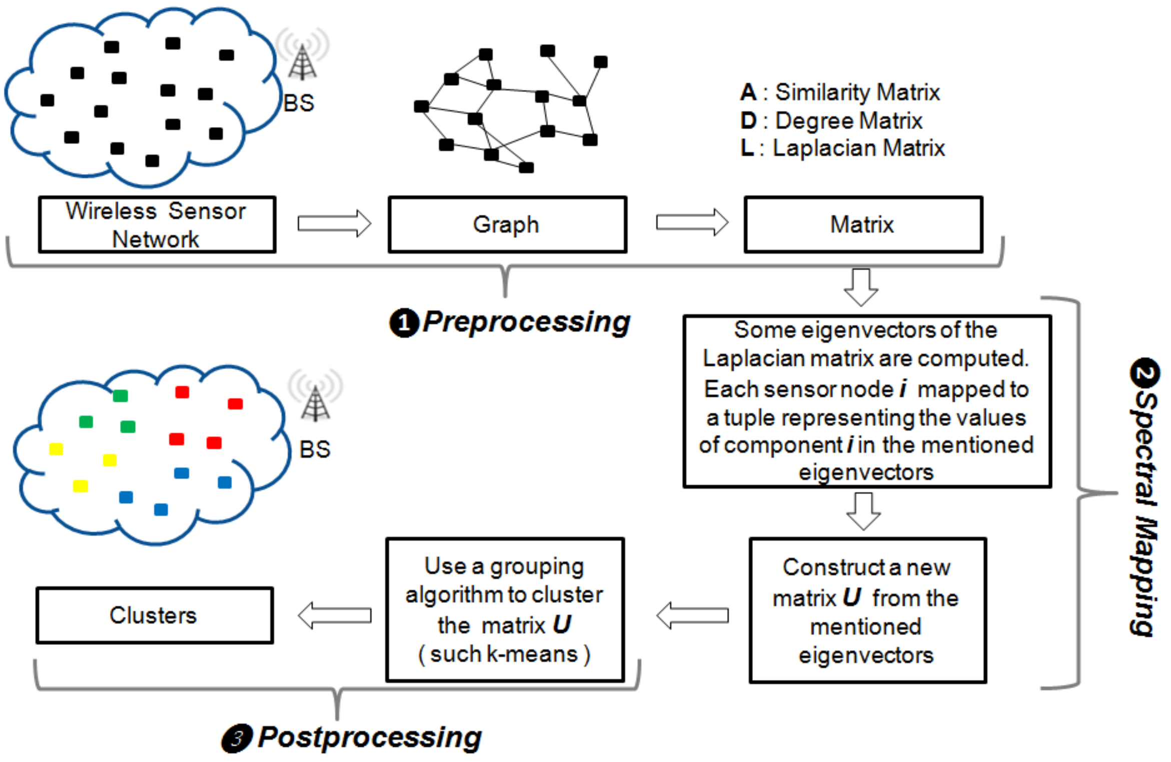

BS
A : Similarity Matrix
D : Degree Matrix
L : Laplacian Matrix
Wireless Sensor Network
Graph
Matrix
1 Preprocessing
Some eigenvectors of the Laplacian matrix are computed. Each sensor node i mapped to a tuple representing the values of component i in the mentioned eigenvectors
2 Spectral Mapping
BS
Clusters
Use a grouping algorithm to cluster the matrix U (such k-means)
Construct a new matrix U from the mentioned eigenvectors
3 Postprocessing

Fig. 3. Spectral clustering scheme of the MR-KSCA approach.

sound waves [22]. After that, it transmits the derived position to the BS in a short message. Based on the spectral clustering principle, the BS constructs the graph G and the similarity matrix A which represents the network. There are several popular constructions to transform a given set of data points with pairwise similarities into a graph. When constructing similarity graphs the goal is to model the local neighborhood relationships between the data points [7]. Here we simply connect all points with positive similarity with each other, and we weight all edges by a_{ij}. In our approach we consider the similarity function as a Gaussian similarity function. Hence, similariy matrix is constructed as follows:

$$A=[a_{ij}]=\begin{cases} exp\left(\frac{-1}{2\sigma^2} * d^2(i,j)\right) & if\ i\neq j \\ 0 & otherwise \end{cases} \qquad (6)$$

with $d(i,j)$ is the Euclidean distance between nodes i and j. It should be noted that a_{ij} is large when points indexed by i and j are likely to be in the same cluster.

Thereafter, the BS must deduce the Degree Matrix and the Laplacian one. Then, it computes the eigenvalues and the eigenvectors of the last matrix.

B. Clustering step

The objectives of the current step are to define the optimal number of clusters and to form them.

As indicated in [15], the optimal number of clusters can be determined as follows:

$$K=\frac{\sqrt{N}}{\sqrt{2\pi}} * \sqrt{\frac{E_{fs}}{E_{ms}}} * \frac{M}{d_{toBS}^2} \qquad (7)$$

where N is the number of sensor nodes, M is the area of sensor nodes deployment, and d_{toBS} is the average distance of the nodes from the BS.

We construct a new matrix U from the K eigenvectors related to the K largest eigenvalues of the Laplacian matrix. In order to determine the K clusters of the WSN, we apply the classification algorithm k-means to the matrix U. We deal with each row of U as a point in $\Re^{K\times N}$. We cluster the WSN into K clusters via k-means. The sensor node i is assigned to cluster C_j if and only if row i of the matrix U was assigned to cluster C_j.

We notice that in the proposed algorithm we determine the clusters before specifying the C-Hs and the optimal number of clusters is as well defined automatically. So, our algorithm is different from the others (such as LEACH, LEACH-C, DECSA, etc.) that determine the C-H nodes before the clusters. In addition, in order to define the number of cluster heads, the latter protocols run the same technique in each iteration and by the way consume more energy.

C. Cluster head election step

Once the clusters are determined, the next step of the MR-KSCA consists to select the C-H nodes. The C-H nodes are responsible of the coordination among the nodes within their clusters, as well as for the aggregation of the received information and the communication with the BS or the super C-H node. In our algorithm, the C-H is determined by taking into account the *id* of the node in the cluster and its residual energy (Algorithm 1). The *id* is defined without considering the position of the node in the cluster. The C-H in each round of communication will be at a random position on the cluster. Thus, the probability of uncovered area will decrease extremely.

Indeed, for each round r of the simulation, we use the number $C_k = (r \ mod\, |S_k|)$ to elect a C-H for the appropriate cluster; $|S_k|$ is the total number of nodes in the cluster k. The node with $id = C_k$ and residual energy E_r greater than threshold E_{rmin} will be the C-H of the cluster k in the round r. E_{rmin} is the minimum residual energy required for a given node to be a C-H. It is the summation of the energy needed to receive and process data coming from the appropriate cluster nodes (data aggregation), and to transmit data towards the BS. This energy E_{rmin} is given by:

$$E_{rmin} = |S_k| \ * (E_{Rx}\,(L, d) + E_{Aggregation}) + E_{Tx}\,(L, d) \tag{8}$$

with

- $E_{Tx}\,(L, d)$is the energy consumed when the C-H transmits $L - bits$ data to the BS by a distance d.
- $E_{Aggregation}$ is the energy needed by the C-H to process data:

  $$E_{Aggregation} = L \ * E_{DA} \tag{9}$$

  E_{DA} is amount of energy for data aggregation.
- $E_{Rx}\,(L, d)$is the energy consumed when the C-H receives data from one node.

Hence,

$$E_{rmin} = L * \big((|S_k| + 1) \ * E_{elec} + |S_k| \ * E_{DA} + \epsilon_{mp} \ * d_i^4\big) \tag{10}$$

Nevertheless, if the residual energy E_r is less than E_{rmin} this node must broadcast a short message informing the node with $id = C_k + 1$to its residual energy and so on. Thus, the energy consumption will be distributed with more equitable between all nodes.

Algorithm 1 Cluster Head Election

```
• For each round $r$
    – For each cluster $k$ in $\{1, 2, \cdots, K\}$
        ∗ Each node $i$ of the cluster $k$ with the $id$ in $\{1, 2, \cdots, |S_k|\}$ computes the value $id_{CH} = (r \ mod\, |S_k|)$
        ∗ While $E_r(id_{C-H}) \leq E_{rmin}$
            · $id_{C-H} = id_{C-H} + 1$
          End While
        ∗ Node with $id = id_{C-H}$ will be the C-H for the current round $r$
      End For
  End For
```

D. Super cluster head election step

This step consists of selecting the super C-H node from the set of C-H nodes. In our approach, the ordinary nodes report their data to the respective C-H nodes. The C-H nodes aggregate the data and send them to the BS through the super C-H node. If the C-H nodes transmit data over longer distance to reach the BS directly, they lose more energy compared to ordinary nodes. Hence, the super C-H node play the role of the second BS. We also assume that if the distance between the C-H node and the BS is lower than the distance between the C-H node and the super C-H node. Then, the C-H node transmits its data directly to the BS.

Thus, in our algorithm the super C-H is determined by taking into account the residual energy of the nodes. The super cluster head will be the C-H with the highest level of the residual energy.

E. Data Transmission

Based on the id and the numbers of nodes in the cluster, the C-H node creates a schedule TDMA to assign to each member's node a time when it can transmit its data to its own C-H. If we suppose that the node with the $id \ = \ i$ is elected as a C-H, the node with $id = ((i + 1 + |S_k|) \ mod\, |S_k|)$ is assigned the first round to transmit its data.

One of the most important challenges in our approach consists in reducing the total consumed energy of each round. Hence, we avoid energy consumption due to synchronization of the cluster nodes when the C-H is elected to assign the TDMA. Indeed, this technique ensures not having collisions between data messages. It also allows the radio components of each non-C-H node to be disabled at any time, except during the time it is allocated for transmitting these data. this reduces the energy consumed by the nodes. However, the C-H must keep its receiver "ON" in order to receive data from other nodes [23]. In addition, to save more energy in a WSN, we assume that if the distance between a node and the BS is lower than the distance between this node and its C-H. Then the node transmits its data directly to the BS. Moreover, each node can move in the standby mode to reduce energy consumption.

Also, to avoid unnecessary node s control messages transmission and control overhead of the BS, the clusters are created only when the sensor nodes are deployed in the first round. So, the calculating overhead is only C-H selecting and super C-H selecting in the most set-up phase.

V. Simulations and Results

In this section, we present the results of the evaluation experiments of the proposed MR-KSCA algorithm. We used the MATLAB software to simulate and analyze this algorithm. Besides, we consider the radio model presented in Figure 1. The different parameters used in our simulations are shown in Table I.

A node that runs out of energy is considered dead and cannot transmit or receive data. For these simulations, the energy of a node decreases each time it sends, receives or aggregates data according to the model radio parameters. Also, we ignore the effect caused by the signal collision and the interference in the wireless channel. Furthermore, each simulation result shown below is the average of 100 independent experiments,

Algorithm 2 Data Transmission

```
• For each round $r$
  – For each cluster $k$ in $\{1, 2, \cdots, K\}$
      ∗ The C-H node creates Time-Slot assignment TDMA.
      ∗ The C-H node collects measurement from cluster's nodes.
      ∗ If the distance between C-H and BS is lower than the distance between C-H and super C-H
          · The C-H node sends aggregated data to the BS.
        Else
          · The C-H node sends aggregated data to the super C-H node.
        End If
    End For
  End For
```

TABLE I. EXPERIMENTAL SIMULATION PARAMETERS.

Parameter	**value**
E_{elec}	$50nJ/bit$
E_{fs}	$10pJ/bit/m^2$
E_{mp}	$0.0013pJ/bit/m^4$
Initial energy E_0	$0.5J$
E_{DA}	$5nJ/bit/message$
Area of Network	200m ×200m
Sink coordination	$(100m, 250m)$
d_0	$88m$
Message Size	$4000bytes$
Number of Nodes	500

where each experiment uses a different randomly-generated uniform topology of sensor nodes.

As illustrated in Figure 4, an example of WSNs with $N = 500$ nodes randomly distributed in a 200m ×200m area. The BS is located far away from the sensing area $(x_{SB} = 100m\,;\, y_{SB} = 250m)$.

In order to evaluate the performances of the new proposed protocol, we propose to compare it to:

- K-way Spectral Clustering Algorithm (KSCA) [1].
- Spectral Classification based on Near Optimal Clustering (SCNOC) [20].
- Distance-Energy Cluster Structure Algorithm (DECSA) [17].
- Low Energy Adaptive Clustering Hierarchy Centralized (LEACH-C) [15].

Also, we use two metrics to analyze and compare the

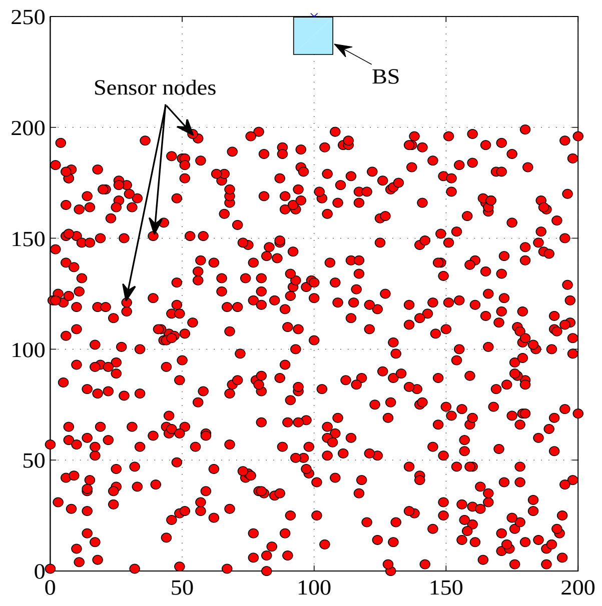


Fig. 4. Random deployment of 100 nodes over an area of size 200m ×200m.

performances of the protocols:

- **Network lifetime:** It is necessary that all nodes stay alive as long as possible. Network quality decreases considerably as soon as one node dies. Thus, it is important to know when the First Node Died. Furthermore, sensors can be placed in proximity to each other. Thus, adjacent sensors could record related or identical data. Hence, the loss of a single or few nodes does not automatically diminish the quality of service (QOS) of the network. In this case, the metric **First Node Died** (FND) denotes the time elapsed until first node depletes its energy. The metric **Half of the Nodes Died** (HND) denotes the time elapsed until half of the nodes in the network are dead. Finally, the metric **Last Node Died** (LND) gives an estimated value for the overall lifetime of a sensor network.
- **Energy consumption:** Uniform energy consumption is very important for network load balancing: More uniform energy consumption, less possibility for node premature death. Less energy consumption per round, better network performance.

As illustrated in Figure 5, the main problem of LEACH-C and DECSA protocols are the random selection of the C-H nodes. It is obvious that a stochastic C-Hs selection will not automatically lead to minimum energy consumption during data transfer for a given set of nodes. All C-H nodes can be located near the edges of the network or adjacent nodes can become C-Hs simultaneously. In these cases some nodes have to bridge long distances to reach a C-H.

In SCNOC algorithm, when we use only the second eigen-

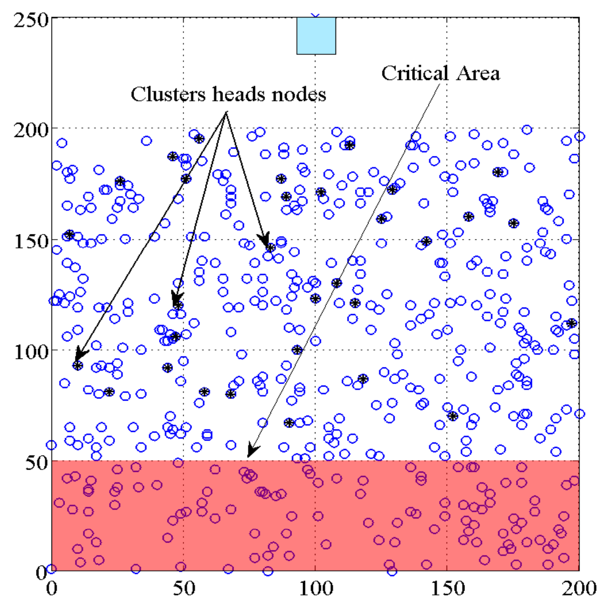


Fig. 5. Illustration of the critical area in the LEACH-C clustering algorithm with 500 nodes.

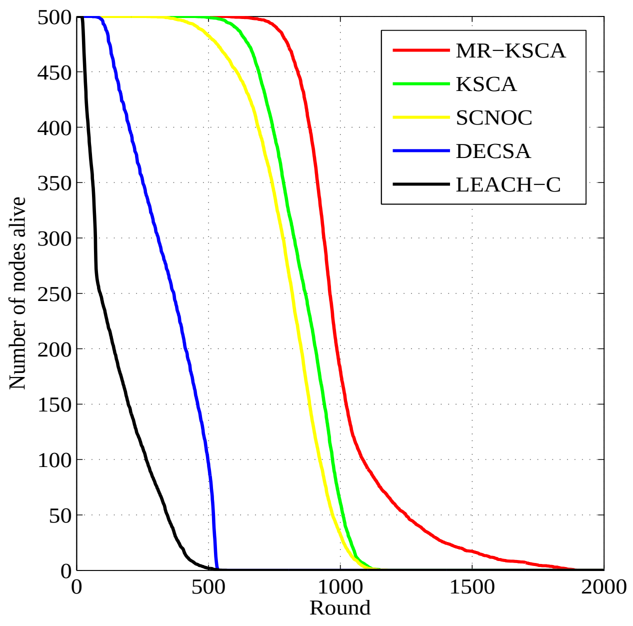


Fig. 7. Number of nodes alive over time of the compared protocols.

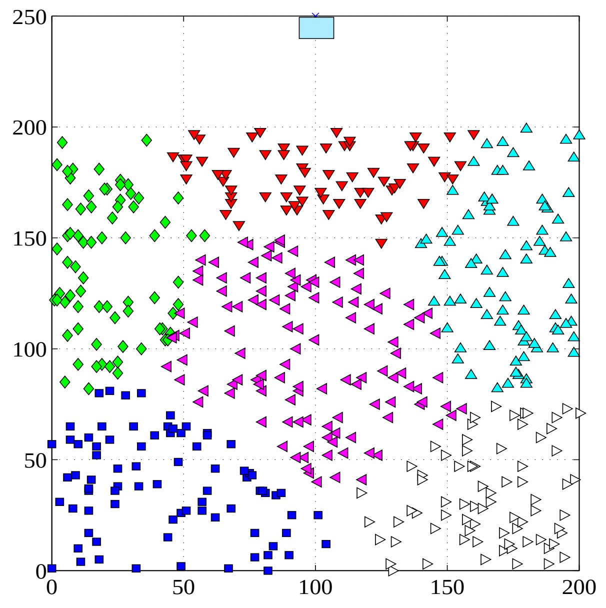


Fig. 6. Clustering results of the MR-KSCA algorithm with 500 nodes.

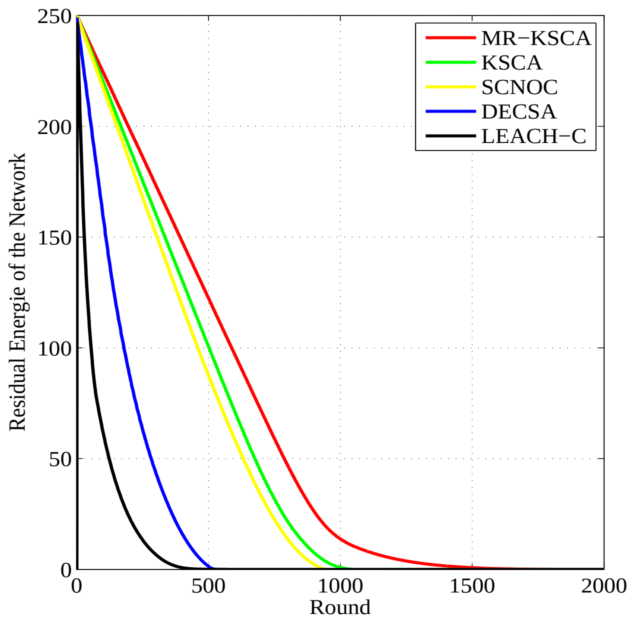
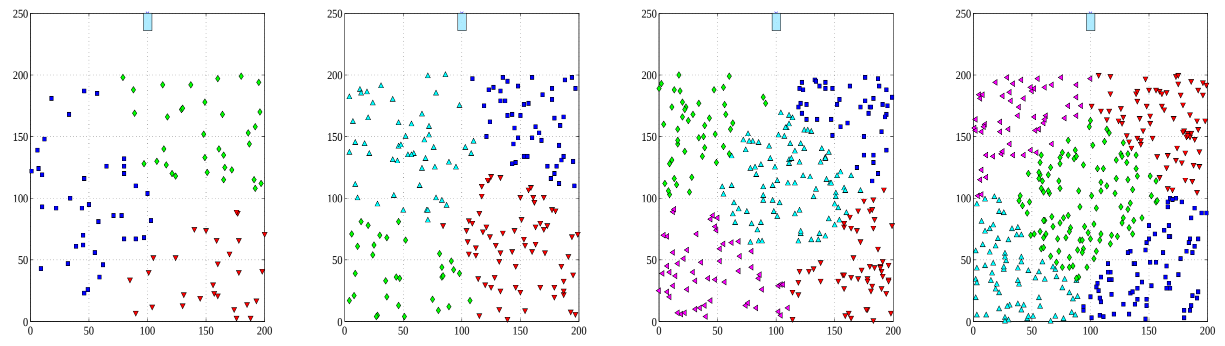


Fig. 8. Evolution of the remaining energy in the network when the transmission rounds.

vector of the Laplacian matrix in the clustering setup, we may lose the connectivity information about the network [11] and obtain a poor clustering setup.Also, we cannot partition the network into the optimal number of clusters. On the other hand, the rotation selection of the C-H nodes may obtain a poor C-H selection setup, the distribution of C-H nodes is not uniform and some sensor nodes have to transfer data through a longer distance. Hence, reasonable energy saving is not obtained.

In KSCA, all C-H nodes transfer their data directly to the BS. Hence, some C-Hs which are far away from the BS consume exhaustive energy.

Figure 6 presents an example of clustering setup for the MR-KSCA. We note that the network is subdivided into



(a) N=100 (b) N=200 (c) N=300 (d) N=400

Fig. 9. Clustering results of MR-KSCA algorithm for different number of nodes.

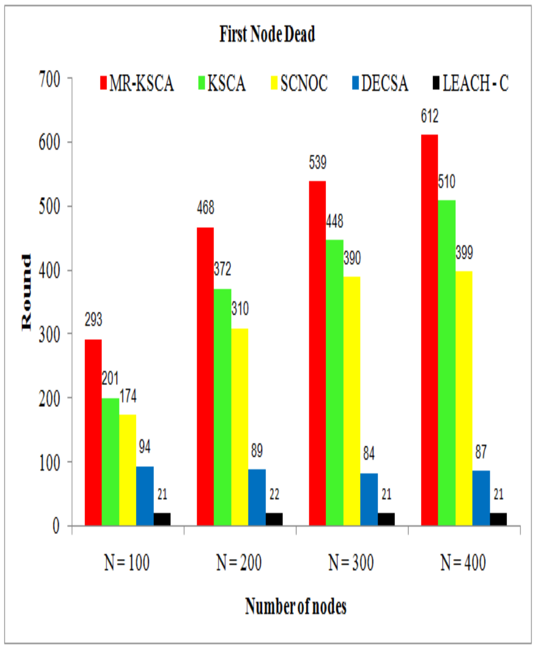


Fig. 10. Impact of the node density on the ”First Node Died”.

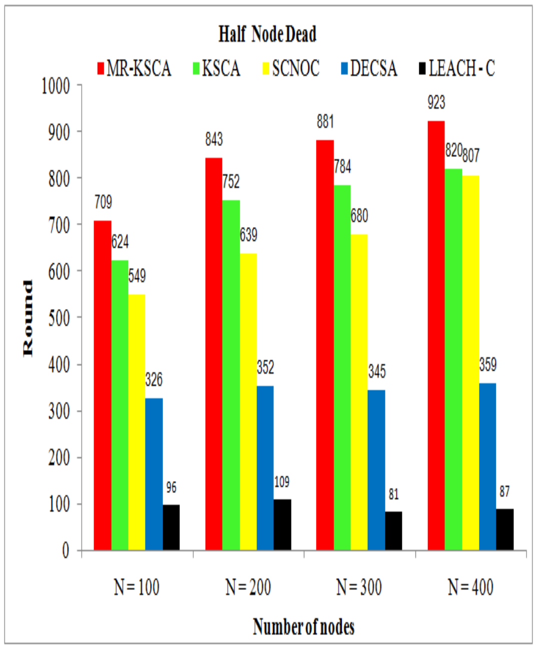


Fig. 11. Impact of the node density on the ”Half Node Died”

six clusters, and the nodes are correctly distributed over the sensing area. Also, there is no intersection between the different clusters.

Figure 7 shows a significant improvement for MR-KSCA in terms of numbers of periods relating to FND, HND, and LND. In the proposed method, FND occurs at the round 641 while it occurs at the round 520 for KSCA, at the round 378 for SCNOC, at the round 21 for LEACH-C, and at the round 74 for DECSA protocols. HND occurs at the round 960 while it occurs at the round 867 for KSCA, at the round 817 for SCNOC, at the round 89 for LEACH-C, and at the round 368 for DECSA protocols. Finally, LND occurs at the round 1865 while it occurs at the round 1171 for KSCA, at the round 1080 for SCNOC, at the round 568 for LEACH-C, and at 541 for DECSA protocols.

Figure 8 gives the total network energy remaining in every transmission round. The energy remaining decreases more rapidly in KSCA, SCNOC, LEACH-C, and DECSA than in MR-KSCA. In the first 500 transmission rounds, approximately 51.3% of the total energy is consumed in case of MR-KSCA. Whereas, 60% in KSCA, 65% in SCNOC, 98% in DECSA, and 100% in LEACH-C.

Figure 9 presents the clustering setup for different number of nodes. In each case, we note that the network is subdivided into an optimal number of clusters. There is no intersection between the different clusters. Also, the C-H nodes will be distributed over the sensing area.

In Figures 10-12, we show the effects of the node density on the compared clustering techniques as well as on the network's stable regions (”FND”, ”HND”, ”LND”). For different values of N equal to 100, 200, 300, and 400, our algorithm presents an improvement of performances compared to the other algorithms. It follows that even if the node density increase the new proposed approach still gives best results compared to the

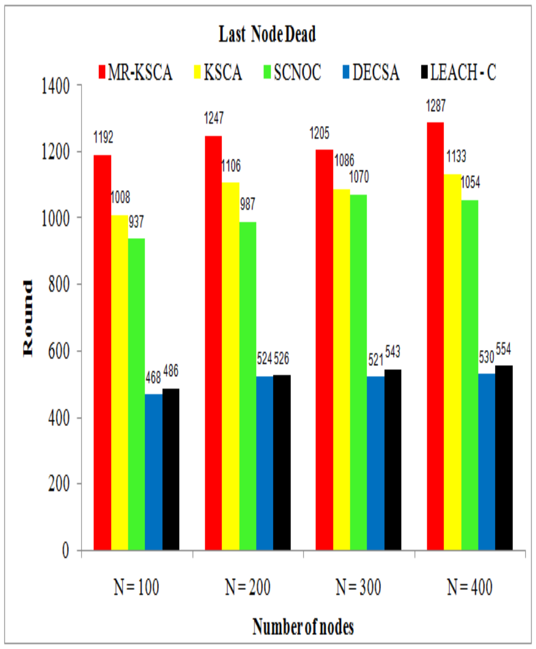


Fig. 12. Impact of the node density on the ”Last Node Died”

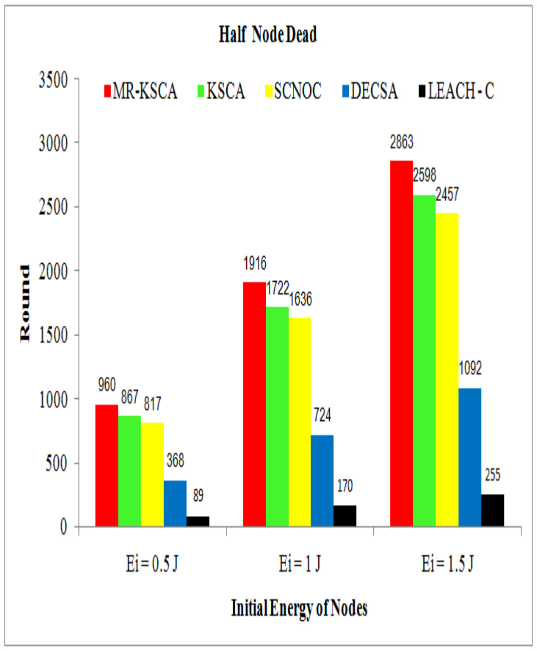


Fig. 14. Impact of the initial energy quantity on the ”Half Node Died”.

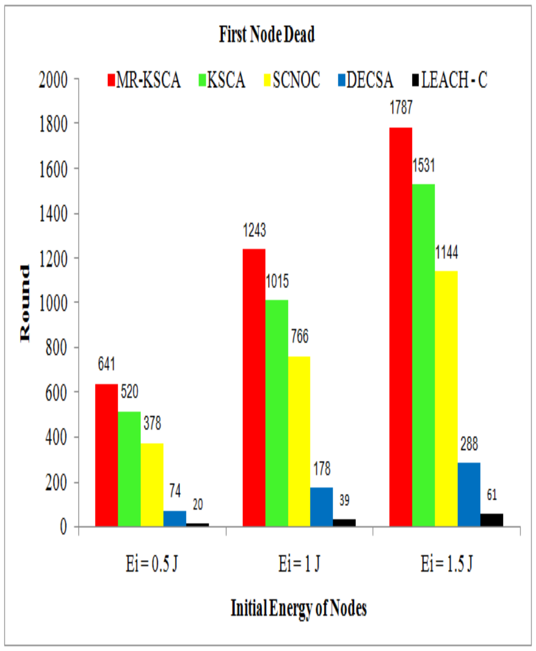


Fig. 13. Impact of the initial energy quantity on the ”First Node Died”.

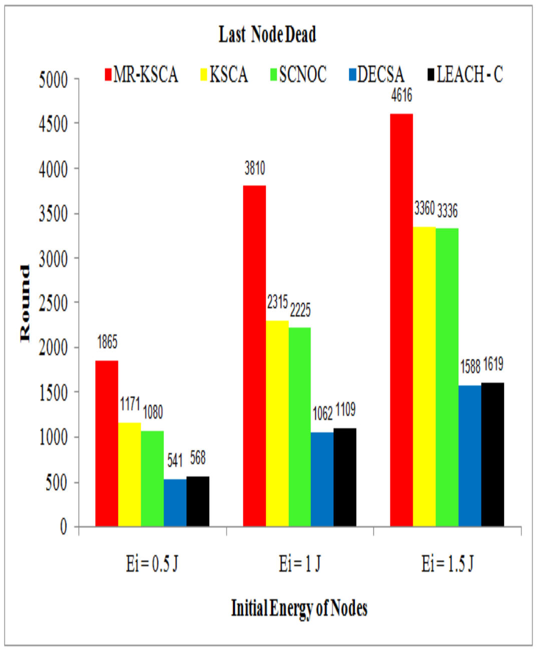


Fig. 15. Impact of the initial energy quantity on the ”Last Node Died”.

other ones.

In Figures 13-15, we show the performances of the different compared protocols by using different initial energies. It gives respectively the ”FND”, ”HND”, and ”LND” depending on the quantity of the nodes initial energy.

Once more, it is shown that for different values of the energy, the new proposed approach presents a significant improvement compared to the others. We conclude that the MR-KSCA algorithm gives a significant performance improvement; in terms of energy and lifetime gains, compared to the KSCA, the SCNOC, the LEACH-C, and the DECSA protocols. The best results of the MR-KSCA approach can be explained by the three points: (i) The proposal starts by selecting the clusters (similar nodes) before the election of the C-Hs. (ii) The approach considers the residual energies of the nodes when we select the C-H nodes. (iii) The C-H nodes are distributed in the WSN.

VI. Conclusion and future work

Energy saving is an important challenging issue in a WSN. To ensure more effective energy distribution and extend the lifetime of the network, energy-efficient saving scheme is developed in this paper. We have proposed a new method of clustering (MR-KSCA), based on spectral classification (K-way) that has been widely used to solve graph partitioning problem. Our approach starts by determining similar nodes (clusters) and after that, selects C-H nodes. Furthermore, MR-KSCA considers the energy of the nodes when selecting C-H nodes and the super C-H node, so the nodes with more energy have more chance to be the C-Hs or super C-H. Hence, balancing energy consumption is guaranted. Indeed, simulation results show that our approach ensures the low energy consumption and improves the network lifetime. Further directions of this study will be deal with clustered sensor networks with more than two levels of hierarchy and more than three types of nodes.

References

[1] A. Jorio, S. El Fkihi, B. Elbhiri, and D. Aboutajdine, “A new clustering algorithm in wsn based on spectral clustering and residual energy,” Proceedings of the Seventh International Conference on Sensor Technologies and Applications (SENSORCOMM), pp. 119-125, 2013.

[2] M. F. Othman and K. Shazali, “Wireless sensor network applications: A study in environment monitoring system,” Procedia Engineering, vol. 41, pp. 1204-1210, 2012.

[3] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, “A survey on sensor networks,” IEEE Communications Magazine, vol. 40, no. 8, pp. 102-114, 2002.

[4] J. Al-Karaki and A. Kamal, “Routing techniques in wireless sensor networks: A survey,” IEEE Transactions on Wireless Communications, vol. 11, no. 6, pp. 6-28, 2004.

[5] M. M. Afsar and M. H. Tayarani-N, “Clustering in sensor networks: A literature survey,” Journal of Network and Computer Applications, vol. 46, pp. 198-226, 2014.

[6] G. Fung, “A comprehensive overview of basic clustering algorithms,” 2001.

[7] U. Von Luxburg, “A tutorial on spectral clustering,” Statistics and Computing, vol. 17, no. 4, pp. 395-416, 2007.

[8] F. R. B. M. I. Jordan and F. R. Bach, “Learning spectral clustering,” Advances in Neural Information Processing Systems, vol. 16, pp. 305-312, 2004.

[9] J. Shi and J. Malik, “Normalized cuts and image segmentation,” IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 22, no. 8, pp. 888-905, 2000.

[10] M. E. Newman, D. J. Watts, and S. H. Strogatz, “Random graph models of social networks,” Proceedings of the National Academy of Sciences, vol. 99, no. suppl 1, pp. 2566-2572, 2002.

[11] A. Ng, M. Jordan, Y. Weiss et al., “On spectral clustering: Analysis and an algorithm,” Advances in Neural Information Processing Systems, vol. 2, pp. 849-856, 2002.

[12] A. A. Abbasi and M. Younis, “A survey on clustering algorithms for wireless sensor networks,” Computer Communications, vol. 30, no. 1415, pp. 2826-2841, 2007.

[13] O. Boyinbode, H. Le, and M. Takizawa, “A survey on clustering algorithms for wireless sensor networks,” International Journal of Space Based and Situated Computing, vol. 1, no. 2, pp. 130-136, 2011.

[14] W. Heinzelman, A. Chandrakasan, and H. Balakrishnan, “Energy efficient communication protocol for wireless microsensor networks,” IEEE Proceedings of the Hawaii International Conference on System Sciences, pp. 1-10 , 2000.

[15] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, “An application-specific protocol architecture for wireless microsensor networks,” IEEE Transactions on Wireless Communications, vol. 1, no. 4, pp. 660-670, 2002.

[16] S. Jerusha, K. Kulothungan, and A. Kannan, “Location aware cluster-based routing in wireless sensor networks,” International Journal of Computer & Communication Technology, vol. 3, no. 5, 2012.

[17] Z. Yong and Q. Pei, “A energy-efficient clustering routing algorithm based on distance and residual energy for wireless sensor networks,” International Workshop on Information and Electronics Engineering, vol. 29, pp. 1882-1888, 2012.

[18] M. Ye, C. Li, G. Chen, and J. Wu, “EECS: an energy efficient clustering scheme in wireless sensor networks,” in Proceedings of IEEE International Performance Computing and Communications Conference (IPCCC), pp. 535-540, 2005.

[19] J.-Y. Chang and P.-H. Ju, “An efficient cluster-based power saving scheme for wireless sensor networks,” EURASIP Journal on Wireless Communications and Networking, vol. 2012, no. 1, pp. 1-10, 2012.

[20] B. Elbhiri, S. El Fkihi, R. Saadane, and D. Aboutajdine, “Clustering in wireless sensor networks based on near optimal bi-partitions,” 6th EURO-NF Conference on Next Generation Internet(NGI), pp. 1-6, 2010.

[21] C. F. Garca-Hernndez, P. H. Ibarguengoytia-Gonzalez, J. Garca-Hernndez, and J. A. Prez-Daz, “Wireless sensor networks and applications: a survey,” International Journal of Computer Science and Network Security, vol. 7, no. 3, pp. 264-273, 2007.

[22] G. Mao, B. Fidan, and B. Anderson, “Wireless sensor network localization techniques,” ACM Computer networks, vol. 51, no. 10, pp. 2529-2553, 2007.

[23] G. Anastasi, M. Conti, M. D. Francesco, and A. Passarella, “Energy conservation in wireless sensor networks: A survey,” Ad Hoc Networks, vol. 7, no. 3, pp. 537-568, 2009.

Peer-to-Peer Collaboration in PeCoCC Framework

Mais Hasan, Robbi Golle, Maik Debes, Jochen Seitz
Communication Networks Group
Technische Universität Ilmenau
Ilmenau, Germany
{first.last}@tu-ilmenau.de

Agnieszka Lewandowska
Staatliche Studienakademie Thüringen
Berufsakademie Eisenach
Eisenach, Germany
lewandowska@ba-eisenach.de

Abstract - **Computer-supported collaborative learning is an important domain of e-learning dealing with researching efficient methods to encourage people to learn together with the help of their computers. Many learning environments support this kind of collaboration and provide several applications to fulfill the needs for efficient learning. These learning environments are usually client-server based solutions, where the users have to access the responsible server to get the services and data they need. This fact leads to two main problems of scalability and single point of failure. Because of its structure, which is similar to the collaborative learning network, the peer-to-peer technology is suggested as a solution for these problems. This paper introduces a framework for using Peer-to-Peer communications, protocols and services in the area of computer-supported collaborative learning. It consists of four layers and includes different Peer-to-Peer modules, which will be selected according to the application requirements. It also provides a messaging system, which uses the overlay-multicasting protocols of the supported Peer-to-Peer modules. This paper shows the advantage of the proposed framework.**

Keywords - Peer-to-Peer communications; computer-supported collaborative learning; Scribe overlay multicasting protocol.

I. INTRODUCTION

During the last decade, online learning has gained enormous interest in most educational institutes. E-learning can be defined as the process of using electronic media, information and communication technologies in education. E-learning includes numerous forms of educational technology in learning and teaching and can be used jointly with conventional face-to-face learning. E-learning can occur in or out of the classroom. It is suited to distance and flexible learning and can be asynchronous or synchronous. As a result of the rapid improvement in the areas of education, information and communication technologies, various e-learning methods have evolved. This evolution started with using the information technology in Computer Based Training (CBT) and developed in the direction of exploiting the Internet and social interaction in Virtual Learning Environments (VLE) and Computer-Supported Collaborative Learning (CSCL) applications.

The most used learning environments have been based on the client/server approach. A server is the source of services and information, several clients have access to. However, the approach suffers from two main problems: scalability and single point of failure. Thus, different approaches have been developed to overcome these problems. One of these is a paradigm shift to the Peer-to-Peer (P2P) model, which offers better load balancing and robustness compared with the conventional client-server model. In this approach, the communication partners act as servers and clients at the same time. They all offer a part of the information and retrieve information from other nodes known as peers. The more peers take part in a P2P network, the better this network scales and the higher its reliability is. Several application fields have utilized this P2P approach so far. In this paper, we introduce a framework to apply this approach to computer-supported collaborative learning.

Therefore, the paper is organized as follows: Section II shortly deals with computer-supported collaborative learning and reviews some CSCL-tools based on P2P technology. Different P2P technologies and their properties are analyzed in Section III. Section IV presents our designed CSCL-tool; the **pe**er-to-peer **co**mmunications for **c**omputer-supported **c**ollaborative learning (PeCoCC) framework and its functioning [1]. A description of the used software and a primary implementation of Scribe overlay multicasting protocol in the P2P simulator PeerfactSim.KOM follows in Section V. Section VI gives an overview of the current state of the work and summarizes the paper.

II. PEER-TO-PEER COMPUTER-SUPPORTED COLLABORATIVE LEARNING

Computer-supported collaborative learning is an emerging branch of e-learning allowing several students to cooperate with each other and with the teaching staff online in order to solve shared tasks or to exchange their skills. Computer-supported collaborative learning is related to collaborative learning and Computer Supported Cooperative Work (CSCW). By collaborative learning, we generally mean that a group of students work together to discuss, solve or evaluate teaching materials; on the other hand, computer supported cooperative work addresses the technologies and tools supporting people in their work. Hence, computer supported collaborative learning refers to the use of CSCW-technologies and tools by a group of collaborative students in a learning process. These technologies and tools have been developed to provide an efficient learning process. Woodill [2] gives an overview of all the different technologies used to support collaborative learning (see Fig. 1).

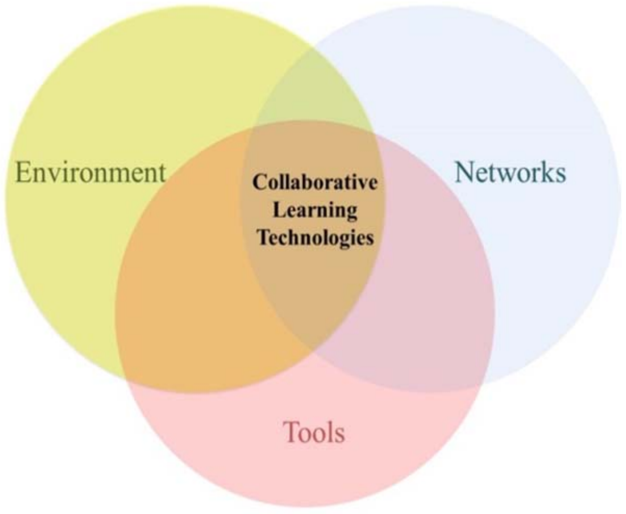


Figure 1. Information technologies used to support collaborative learning [2]

Generally, collaboration among the students is accomplished by many scenarios, which offer team work and the distribution of teaching materials among the users themselves. People participating in such teams need to be able to access the offered resources and services regardless of their current location. They also need effective means to communicate and cooperate with each other. In general, we can recognize three collaborative scenarios according to three attributes, the time, the place and the tasks of the collaboration. The first scenario can be called ad-hoc scenario, which provides collaboration in a limited geographical area and enables the users to interact and communicate with limited collaborative functionalities. Typical examples are conferences where people meet, exchange contacts and ideas, but do not work together on a shared task. The second possible cooperative scenario is the short-term collaboration, which covers a limited time period and enables the group members to collaborate in order to achieve a shared task, e.g., work on a paper, or integrate software components. Such collaboration requires access to shared resources as well as knowledge and data exchange. It is also based on a trust relationship among the members, who do not have to be in the same place. The third scenario presents the long-term collaboration, which is used by teams for longer time periods, e.g., within a project. In this case, it is necessary to offer flexible and different interaction and cooperation possibilities and services. Usually, a centralized infrastructure is used for this scenario.

In all these scenarios, the user must be able to find and access the required resources and services. In this case an efficient distributed search based on the semantic description of the according data, users, services, etc. is needed. Moreover, distributed storage and a secure access to the resources are really important in these collaborative systems. Furthermore, a team membership management service, which should be associated with a trust model as well as a cooperative messaging scheme among the users, is necessary to provide a flexible interaction among the group members [3].

P2P systems as one of the technologies, which are used in the collaborative systems, provide the efficient, scalable distributed search, while the semantic search in P2P networks embodies one of the important current research areas. Some more research work has been done to provide distributed storage and cooperative messaging to some P2P technologies. Furthermore, educational P2P applications like, e.g., COMTELLA, EDUCOSM, Edutella, and Groove have been developed for some specific needs and they are still under development. A brief description with a comparison between these applications will be highlighted in the following.

COMTELLA is a P2P file sharing system that allows students to contribute and share class-related resources with their community [4]. The shared papers are annotated with respect to their content in categories. COMTELLA uses a modified version of the standard Gnutella P2P protocol and instead of sharing the actual files, only their URLs are shared. A list of the shared articles, their URLs in the web and the corresponding comments are distributed among the users. There is one list for every category. If a student searches for a paper, he should only search the list of the matching category. Students can view and read the papers without downloading them by clicking on the "Visit" button in the COMTELLA user interface, which starts the default browser with the URL of the paper.

EDUCOSM is a web-based learning environment providing a shared view to the Web [5]. It consists of a collection of server-side scripts and an HTML and JavaScript based client that runs inside a web browser. The role of the server is to store the data and act as a proxy between the client and the rest of the web. The principles of EDUCOSM and COMTELLA are similar with the difference that the storage of the data in COMTELLA is distributed among the users.

Edutella is an educational P2P network built on Sun Microsystems JXTA Framework [6]. Edutella is an open source P2P application for searching semantic Web metadata. It uses the resource description framework (RDF) for presenting information in the web. Edutella deals with metadata about content, not with content itself. It adds a search service to the JXTA framework, so that any node that carries metadata about some resource can announce an Edutella search service to the network. The nodes in Edutella have at least one of three types of roles: provider (provides a query service), consumer (asks questions) and hub (manages query routing in the network).

The previously mentioned three P2P collaborative systems offer only one collaborative tool, mostly a file-sharing application, which is not sufficient for efficient collaboration among the users. These systems suffer the absence of a coordinative tool like a group calendar, which is typical for team or group software. They also do not support cooperation applications like a whiteboard or a text editor.

These problems have been tried to be solved in one of the popular collaborative environments, Groove. It is a collaborative groupware based on the principle of a shared workspace [7]. Tools like a shared browser, a shared drawing board or a file archive are used to operate in this shared workspace. Groove provides servers that are used to detect new peers in the network and to store content if one or more peers are offline and cannot see the changes made at that time. Using server-based services threatens the availability of these services if one of these servers fails.

Groove is targeted at small workgroups and has its own protocols. It is only available for the windows platform, so it suffers interoperability problems. Table I presents a brief comparison between the previously mentioned collaborative systems.

TABLE I. A COMPARISON BETWEEN SOME P2P COLLABORATIVE SYSTEMS

P2P based CSCL Applications	Pure P2P Connectivity and Services	Collaborative Applications	Coordinat-ion and Awareness	Interoper-ability
COMTELLA	+	o	-	+
EDUCOSM	-	o	-	+
Edutella	+	o	-	+
Groove	-	+	+	-

The comparison is based on four attributes, which reflect the flexibility and the effectivity of any P2P collaborative application. The first attribute represents the usage of pure P2P communication and services, which is supported by COMTELLA and Edutella, while the other systems need the functions of servers to provide some services like saving the profiles of users or offering the collaboration space awareness.

The second attribute embodies the effective support of collaborative applications, which is performed very well in Groove, while the other systems offer only one application like a file sharing application or a semantic search application, which is not enough for an efficient collaboration among the users. Unlike Groove, the other three systems do not support any collaboration space awareness or coordination among the users, which is an important attribute to improve the efficiency of the collaboration. The fourth attribute is the availability for different operating systems or in other words the interoperability, which is considered only in COMTELLA, ADUCOSM and Edutella.

The previous comparison manifests the need for a collaborative platform providing many collaborative and coordinative tools, supporting interoperability and work space awareness, and basing on fully distributed server-independent P2P communications and services. However, there is no open source software having the mentioned functionalities available at the moment.

To be able to understand the need of the previously mentioned structures for efficient using of P2P networks in a collaborative scenario, an overview on the P2P technologies with some examples is highlighted in the following section.

III. PEER-TO-PEER TECHNOLOGIES

In contrary to the client-server model, all the members of a P2P network are equally offering and requesting services. Generally, we can assert that every P2P network is established on an overlay network, mostly based on Transmission Control Protocol (TCP) or on Hypertext Transfer Protocol (HTTP) connections. Thus, the overlay and the physical network can be separated completely from each other. Hence, the overlay connections do not reflect the physical connections. Nevertheless, it is possible to match the overlay to the physical network if necessary. P2P networks can be divided into two classes: structured and unstructured P2P networks.

A. *Structured P2P networks*

In a structured P2P network, the network topology and the location of content is determined by employing a P2P protocol. In these networks, the content and the participating nodes share the same address space, which makes it easy and expeditious to reach any content in this space. Structured P2P networks are based on a Distributed Hash Table (DHT) and have no central entities. Frequent signaling traffic is necessary to maintain the network awareness of the nodes [8]. Pastry, Chord and Content Addressable Network (CAN) are examples for this class, two of them, which are used in the PeCoCC framework will be briefly highlighted in the following.

Pastry is a distributed hash table (DHT) algorithm that empowers Internet nodes to build a structured P2P overlay network. It serves as a basis to build up distributed network applications on the top of it [9]. Additionally to its IP address, a Pastry node possesses a unique 128-bit nodeId. It is randomly assigned from a circular nodeId space, ranging from 0 to 2^{128}-1. The assignment is performed in a way, so that nodeIds are uniformly distributed among the nodeId space. As a result, nodes with adjacent nodeIds differ in characteristics such as geography, network attachment or ownership. Routing in Pastry is performed with the help of key values. A corresponding command looks as follows: route (msg, key). A message msg will be routed to the (still alive) node with nodeId numerically closest to the key value. Assuming N nodes are alive within a network, Pastry routes as described within $log_{2^b}(N)$ steps. Delivery of a message is guaranteed when less than $L/2$ nodes fail at the same time. L and b are configuration parameters, where typically $b = 4$ and $L = 16$ or 32. Every key and nodeId in Pastry is a sequence of digits with base 2^b. During a route step, a pastry node routes the message to another node whose nodeId has a prefix, which is one digit ($2^b bits$) longer than the prefix the key shares with the current node´s nodeId. If there is no such node, the message will be forwarded to a node, whose nodeId shares a prefix with the key as long as the current node´s, but numerically closer to the key than the nodeId of the current node. If still there is no such node, then the message has arrived at its destination. Fig. 2 shows a simple routing example [10].

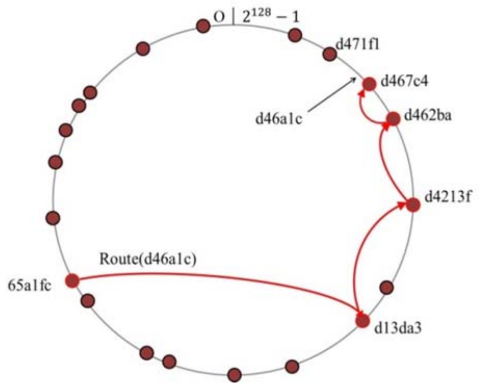


Figure 2. Message routing in Pastry nodeId space [10]

The routing tables of Pastry nodes contain (2^b-1)* $log_{2^b}(N)$ entries. Each entry maps a nodeId to the associated node´s IP address. In the case of node arrival, departure or failure, the routing tables needs to be updated. That can effectively be done by exchanging $O(log_{2^b}(N))$ messages. In addition to the routing table, a node supervises a leaf set. In this leaf set, the node stores the $L/2$ numerically closest larger and smaller nodeIds. Within any network type, inconsistences, like node´s joining, leaving or failing, happen. These inconsistences have to be considered by maintaining the routing table and leaf set of nodes. Special messages are exchanged when some of these events happen, so that the nodes of a pastry network maintain up-to-date routing tables and leaf sets.

Since the PeCoCC framework uses Scribe, which is one of the important Pastry applications, a short overview is introduced in the following paragraph.

Scribe has been defined by its developers as a “scalable application-level multicast infrastructure built on the top of Pastry” [11]. With its help, Pastry nodes can create groups and exchange multicast messages among group members. It is possible to create, join and leave multiple groups. Scribe is able to maintain large amounts of groups with possibly large amounts of group members.

If a node wants to create a group, it determines a group name string, which is mostly the topic of the shared task of the group, and calculates a hash value from it. The groupID is created by concatenating the hash value of the group name and the nodeId (which already is a hash value). The creating node may furthermore decide on group credentials which determine the node characteristics that have to be fulfilled in order to allow this node to join the group. The create message will be sent using the route method of Pastry, where the key in this case is the groupID. The create message will be delivered to the node with nodeId numerically closest to the groupID. This node that is now called the root of the group, then adds the groupID to a list of groups it already knows about.

If a node wants to join a group, it routes a join message through the network. The key of the route method is in this case also the groupID of the group the node wants to join. At each hop in the network, Scribe´s forward method will be invoked. If a node receives the join message, it will look up its groups table if it already knows the group. If not, it adds the group to its groups table. It also terminates the join message and routes a new one on its own, again, with the groupID as a key. Furthermore, it adds the nodeId of the sending node to its children list of the group. This procedure is repeated at the next hops until the message arrives at the root of the group, which in turn adds the nodeId of the last node to its children table of the group.

By terminating the join message and routing a new one at each hop, all nodes will store only the nodeId of the previous nodes in their children lists of groups. This behavior will create a tree-like structure to which multicast messages can be delivered.

If a node wants to distribute a multicast message to a certain group, it routes the message to the root of the group. The message will arrive at the root, which will first deliver its IP address to the sender of the multicast message. In this way, the sender can use the IP address of the root to directly send multicast messages to it in the future, which saves repeated routing. Now, the root sends the message to all the children of the group it is aware of (which are only one hop away). These children will then route the message to the children of the group they are aware of. This behavior is repeated until the message arrives at the leaves of the multicast tree. These leaves are the members of the group. The nodes along the multicast tree are called forwarders of the group, as they forward multicast messages through the tree. Forwarders can also be members of the group. Fig. 3 illustrates a multicast group, which has a root with nodeId (1100) that is numerically closest to the groupID (in this case 1100 too) [12].

If a node wants to leave a group, it locally deletes its nodeId from its children table of the group. If the children table of that group is empty, it sends a leave message to its parent in the multicast tree (the next hop of a route message). The parent deletes the sender’s nodeId from its children table of the group. If the children table is empty, the parent sends a message to its parent nodes. This behavior is repeated until the leave message arrives at a node, which has more than one entry in the according children table.

The second used P2P overlay is the Content Addressable Network (CAN), which is also a structured P2P network based on the distributed hash table concept [13]. The key space in CAN is organized as a virtual d-dimensional torus with Cartesian coordinates. This coordinate space is completely logical and has no relation to any physical coordinate system. It is partitioned among the participating nodes, so each node is responsible for an area of key identifier space, called a zone. This node also maintains information like a coordinate routing table with IP addresses and a virtual coordinate zone of each of its adjacent neighbors in the coordinate space.

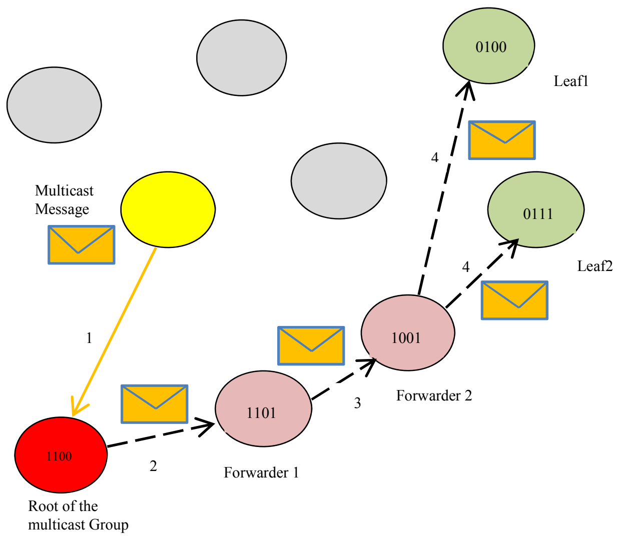


Figure 3. Sending a multicast message to the group with a groupID (1100)

The coordinate space in CAN is used to save (*key*, *value*) pairs by mapping the key onto a point *P* in the coordinate system using a uniform hash function. The corresponding (*key*, *value*) pair is then stored at the responsible node of the zone within which the point *P* lies. To retrieve any entry relevant to any key, any node can apply the same hash function to map that key onto point *P* and then retrieve the corresponding value from the responsible node of the point *P*. A CAN message includes the destination coordinates. Using its neighbor coordinate set in its routing table, a node forwards a message to the neighbor with coordinates closest to the destination coordinates. For a d dimensional space divided into n zones, the average routing path length is $(d/4)*(n^{1/d})$ hops and every node maintains $2d$ neighbors. Taking into account that many different paths exist between two points in the space, a node can automatically route along the next best available path, if one or more of its neighbors failed. One method to improve the performance of CAN includes using multiple hash functions to map a single key to several points of the coordinate space and hence increase the availability. Another method to reduce the routing delay in CAN is maintaining several independent coordinate spaces called realities.

B. Unstructured P2P networks

The distribution of nodes and content in unstructured P2P networks is executed randomly. The position of content can only be resolved in a local proximity of a node and only by flooding the request to a particular extent. In this way, these networks consume the bandwidth, which has been saved by their random distribution. Unstructured P2P networks can be centralized with an index server like Napster, hybrid with dynamic super nodes like Gnutella 0.6 and JXTA, or pure without any central entities like Gnutella 0.4 and Freenet [8]. The following paragraph gives a brief overview of the Gnutella 0.4 protocol, which supports pure P2P connection and presents the simplest form of P2P function in unstructured networks.

Gnutella 0.4 is a decentralized Peer-to-Peer file sharing protocol. Every node in a Gnutella network performs the tasks of client and server at the same time. For this reason these nodes are referred to as servents and the Gnutella protocol defines the way in which these servents communicate over the network. The Gnutella protocol uses ping message flooding to discover hosts on the network. If a servent receives a ping message, it has to respond with one or more pong messages, which include the address of a connected Gnutella servent and information regarding the amount of available data it is sharing on the network. Once the address of a connected servent on the network is obtained, a TCP/IP connection to this servent is created. Thereby, a new servent will be able to take part in a Gnutella network. If one of the servents wants to search for file-based data in the network, it sends a query with some search criteria to its neighbors, which forward this query to their neighbors too, if they do not have the researched file. The forwarding of the query will be continued till the file-based data is found or the time to live field (*TTL*) in the query, which will be decreased by every hop, is found to be zero. If a servent has the research data, it responds with a QueryHit message, which contains enough information to acquire the data matching the corresponding query. This information includes the IP address and the Servent Identifier of the responding host as well as the port number on which the responding host can accept incoming connections and the result set, which identifies file name, file size and file index in the responding host [14].

The servent that receives the queryHit as an answer of its query will initiate a direct download of the requested file, i.e., a direct connection between the source and the target servents is established in order to perform the data transfer. To solve the problem that can arise when the responding servent is situated behind a firewall blocking incoming connections to its Gnutella Port, the servent, which received the queryHit sends then a push message to the servent that sent the queryHit and is supposed to have the researched file. The receiver of the Push message should try to establish a connection to the requesting servent, identified by IP address and port number included in the push message. If the requesting servent is behind a firewall, a connection in the context of Gnutella protocol is not possible.

IV. THE PECOCC FRAMEWORK

The important properties of P2P technologies make them an attractive searching approach to improve performance of some collaborative systems. Until now, most P2P collaborative environments are developed for specific needs and a central entity is used in most of them. Therefore, we have developed a P2P framework for computer-supported collaborative learning, which we called PeCoCC. The PeCoCC framework has been designed to be used in short-term scenario, which serves a limited number of users within a limited time period. It uses different P2P overlays to support different applications. This characteristic of the PeCoCC framework enables completely separate working of the applications, which increases the robustness of the system. The PeCoCC framework has a layered architecture depicted in Fig. 4 [1].

According to the requirements of an efficient collaboration in the short-term scenario, the main services that this framework has to provide are as follows:

Collaborative Tools – The PeCoCC framework provides three applications, which are important to cooperate efficiently. A shared calendar will be used to allow the users to organize their regular meetings; a distributed text editor can be used to jointly make notes on a given subject or to brainstorm about a topic and P2P file sharing allows users to access the distributed contents they need to cooperate.

Session Management Service – Participants in a particular application session, e.g., text editor session, have to be able to get a list of the other members in this session. In this case, a session management unit in the application layer saves a profiles-list of the registered participants for every provided application. This list contains the UserIDs as well as the registered role and priority of every user in the corresponding application. A session management unit also maintains the last case of the application and implements an "Initiation" mechanism to consistently provide the information for latecomers to enable them to participate in the ongoing session. This is typically achieved by getting the state of the distributed application from the current participants and by initializing the application of the latecomer with this information.

Membership Management Service – To be able to use the framework, the user has to be identified by the membership management unit. This unit has to be associated with a trust model, which provides every registered user with an application-ID number (APP-ID number). This number identifies the message traffic in the framework, so that only the registered user can send legal messages.

Repair Mechanism – If a participant in a session suddenly or with intent is going offline, or if the connection of a session member is broken because of technical problems, this mechanism has the task to repair the overlay topology of the network, so that the ongoing session should not be disrupted. This mechanism covers also a late join service, which maintains the overlay topology in the case of newcomers.

Synchronization – For some application modules (e.g., distributed text editor), the group members need to be synchronized to interpret the events in the correct time and order. Furthermore, the most repair algorithms are based on the estimate of the network delay.

Security – As it is mentioned previously, the PeCoCC framework provides many collaborative tools, which based on the exchange of knowledge and data. Therefore, a reliable security mechanism, which controls the participation of a session and takes into account the individuality of the data and documents, is needed. Also, the framework provides security mechanisms (e.g., encryption) to keep personal data secure.

Group Management – Beside the membership management and the session management services, the framework must include functions to manage the communication among collaborative group members in the overlay network. These functions are achieved in the PeCoCC framework using the concepts of overlay multicasting and the cooperative messaging scheme among the peers in the overlay topology.

PeCoCC Messaging System – The PeCoCC framework uses its own message scheme and identifies thereby its used massages in the network and between the layers of the framework. Having an own messaging scheme makes it possible for one user to be member in more than one session and one application. It helps by connection between two overlays, which a user is member in.

The PeCoCC framework also presents the concept of combination between a specific application and the most appropriate P2P overlay for this application. This concept will improve the robustness of the overall system and secure the availability of the other applications in case of attack of one overlay by some possible malicious peers.

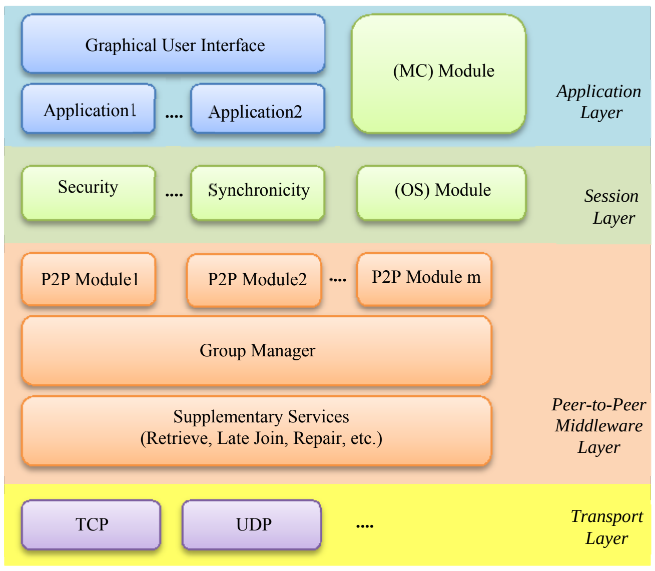


Figure 4. The PeCoCC framework [1]

Therefore, the PeCoCC framework consists of four different layers, which provide its main functionalities and services. The layers are introduced in the next subsections.

A. *Application Layer*

The application layer consists of three main parts. The graphical user interface allows the interaction between the user and the framework. It facilitates a consistent operation of all the desired CSCL services. The PeCoCC Management and Control (MC) module is responsible for controlling the data flow through the framework and the work flow among the users. It embodies the user's membership management and the session management services in the PeCoCC framework. The MC module influences the application program and also saves a list of the cooperating participants, their application-dependent roles and their priorities. The peers should be identified by the MC module to be allowed to enter the system. The rights of the users can be defined by the applications themselves. In a file sharing application, for example, all the users have equal rights, while in a text editor; the teacher should have more possibilities to manipulate the entered information than the students. Furthermore, in sessions without a dedicated chair, the MC module is responsible for defining the user that has the according rights of a chair.

The third part is composed of application modules. These can be freely chosen and added on demand. As stated above, the first modules of our choice are a distributed text editor application, a calendar, and a file sharing application. These modules interact with the graphical user interface and the MC module.

B. *Session Layer*

The session layer provides general mechanisms that are necessary for the offered applications. Currently, we have concentrated on two mechanisms. The security module is responsible for securing private user data and the synchronicity module provides a mechanism to synchronize the different group members so that all of these receive the events in the same order. This service is necessary for real time applications like a distributed text editor. Since the usage of a P2P overlay in the PeCoCC framework is application dependent, the session layer includes the Overlay Selection (OS) module, which is responsible for saving the information about the appropriate P2P overlay for each application. This information will be obtained by several experiments according the latency and load of the network taking into account the requirements of each application.

C. *Peer-to-Peer Middleware Layer*

As P2P technologies exist with respective advantages and disadvantages, the PeCoCC framework allows the usage of different pure P2P technologies. Each technology is encapsulated in a P2P module and offers its communication functionality. Which module should be used is selected by the OS module in the session layer according to the needs of the application. To illustrate the functionality of this layer, we have started with two well-known P2P approaches.

The first overlay is the content addressable network (CAN), which can be used for an efficient distribution of information and teaching materials. As it is mentioned in Section III, it is a structured P2P network based on DHT. CAN offers high scalability and reliability and provides more load balancing than any other pure P2P overlay [15], but it does not take into account the underlying network conditions. Therefore, it is not suitable for real-time applications due to the fact that it does not make any correlation between the overlay distance and the actual number of unicast hops between the hosts in the underlying network.

The second overlay is the structured P2P network Pastry, which considers the underlying network topology and supports a scalable and distributed object location and routing in application layer [8]. Pastry provides an overlay multicasting protocol and collaborative messaging scheme, thus it can be integrated in a P2P module for applications like a distributed text editor and instant messaging.

The Peer-to-Peer Middleware layer also comprises a set of supplementary services that extend the P2P modules with late join, retrieval and repair functions

Furthermore, in each overlay network, the users belonging to one user group have to be managed. This is done in the module called group manager. This module is responsible for forming and supervising a collaborative user group in the overlay network. It uses the concept of overlay multicasting to achieve these tasks.

Overlay multicast implements a multicast service at the overlay network layer. Peers participating in a multicast session use the routing and messaging structure of this overlay. Overlay multicast is achieved through message forwarding among the members of the multicast group at the overlay network using unicast across the underlying network or Internet. So the hosts in overlay multicast handle group management, routing, and tree construction, without any support from the conventional Internet routers, which makes overlay multicasting much easier to deploy than IP multicasting [16].

As Pastry and CAN have been selected as first two P2P overlays in the PeCoCC framework, two overlay multicast protocols built on the top of these overlays have been investigated. CAN-based multicast uses the flooding of messages to all the participating nodes in the group. The members of the multicast group first form a group-specific "mini" CAN and then multicasting will be achieved by flooding over this mini CAN [17]. Scribe is a tree based overlay multicast mechanism built on top of Pastry. It builds a single multicast tree for the whole group. Every tree has a root that is responsible for distributing the message among the multicast tree, which is created by combining Pastry paths from each group member to the tree root [11]. The implementation of our work has been started with Scribe protocol, which is described in more detail in the following section.

D. *Transport Layer*

The transport layer embodies the known transport layer in the OSI model, which provides access to different commonly used transport protocols. In PeCoCC framework, an appropriate transport protocol will be selected accordingly to the requirements of the opened applications. For example, the distributed text editor utilizes the Transmission Control Protocol TCP, which provides a reliable transport service.

E. *Functioning of the System*

To understand the functioning of the PeCoCC framework, a collaborative scenario will be presented in the following.

A user starts an application and wants to create a collaborative session with other users, who he knows. The peer then, which is used by this user, will send an announcement with a defined groupID, which can be a hash value of the topic of the session, in the network. In this way, the other peers will be able to find it and join the collaborative session or group. The users how join a collaborative group, receive from one of the group members the all information, which is needed to be able to interact in the ongoing session (e.g., UserIDs of the other group members, the current state of the opened application and its corresponding data, etc.). Every user has its role and priority, which are registered in its profile in the framework. Only the user with an administrator role can change these registered data. According to this scenario, the functioning of the framework will be highlighted in the following.

When the user starts one of the available applications (e.g., a distributed text editor), the MC module is activated and sends a CHOOSE message to the OS module in the session layer containing information about the opened application. The OS module then decides on the basis of the opened application which P2P module is more appropriate for this application and replies to the MC module in the application layer with an OVERLAY message. The OS module also activates the necessary services for the opened application.

The MC module receives the OVERLAY message and retrieves the saved list of the expected participants (the participants of an application should be previously registered by the MC module and saved in a specific list). The MC module then sends a START message to the corresponding P2P module in P2P middleware layer. In the P2P middleware layer, the selected module starts the P2P connection and takes part in the P2P network. The group manager sends a DISCOVERY message with a group ID to find out if the collaborative group with group ID in the P2P network has already been built or not. If it receives a positive answer, the group manager then sends a JOIN message with the group ID using the overlay multicasting protocol to join a collaborative group and retrieve the information about the participating peers as well as the important data to interact in the current session. This case is illustrated in Fig. 5.

The information retrieved by the group manager is returned to the MC module to specify the role and the rights of the peer in the session. The role of the peer will appear in form of active or inactive interaction possibilities in the user interface.

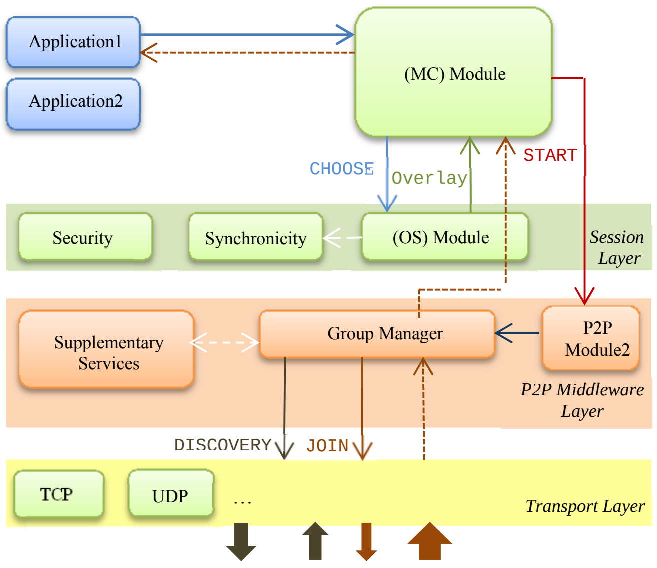


Figure 5. Joining the already built session group

If a collaborative group does not exist, the group manager will send a CREATE message in the P2P network to create a collaborative group with a specific group ID. It will also wait for any join message sent by other peers, which want to join the group. Fig. 6 shows the whole process in this case.

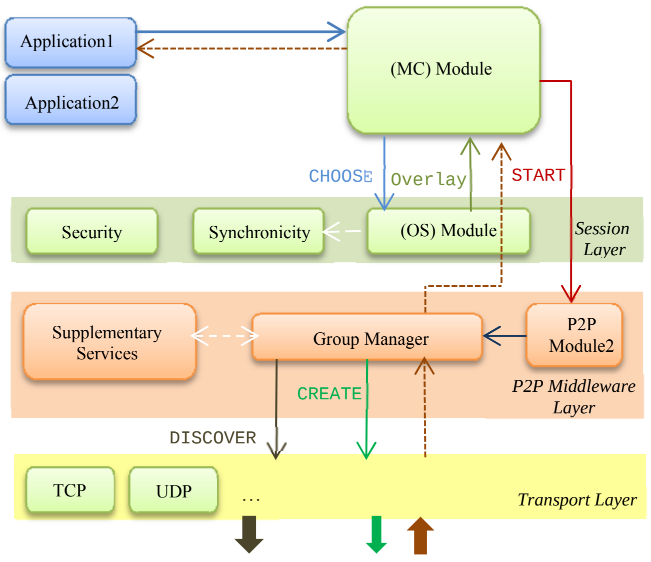


Figure 6. Creating a new session group

Which rights and roles the user has, is managed in tables and saved in the MC module. One table is defined for each application. In the case where all users have the same rights, a mechanism to manage and identify the roles will be used.

This mechanism can take into account the registering sequence of the participants or the alphabetic order of the user ID names, etc.

V. SOFTWARE AND IMPLEMENTATION

The PeCoCC framework is still in implementation phase. It supports the interoperability and is being implemented in Java to be independent from the underlying platform. Three P2P simulators have been studied to select the most suitable one, which cover the main structures of our framework. These simulators are FreePastry [18], PeerSim [19] and PeerfactSim.KOM [20]. These three simulators have been chosen because they are popular among the P2P research community. Furthermore, all three are implemented in Java, the programing language of our framework. In this work, the PeerfactSim.KOM simulator is used because of its visualization capabilities and detailed documentation, which are not available in the other both simulators. In this section, we give a brief overview of the used PeerfactSim.KOM simulator and present the implementation of one of the missing structures in this simulator, which is necessary for the evaluation process of our framework.

A. *PeerfactSim.KOM*

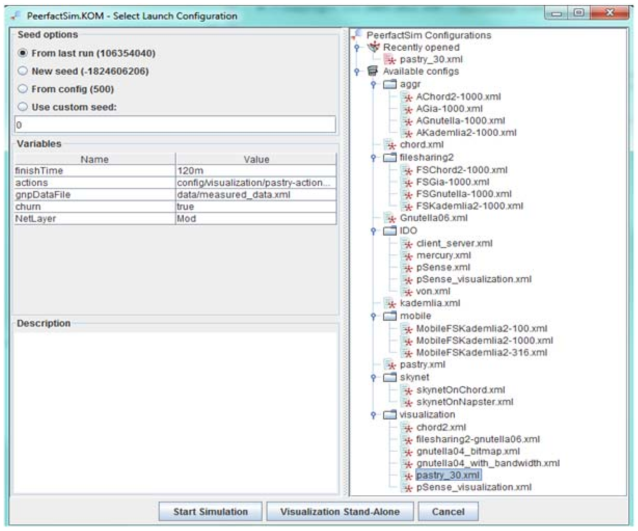
PeerfactSim.KOM is a stand-alone simulator created for simulating P2P Networks. It is a purely event-based simulator and has a layered architecture. This layered architecture is similar to the layered structure of the PeCoCC framework and comprises an application layer, a service layer, an overlay layer, a transport layer and a network layer. At the network layer churn, jitter and latency can be modeled and adjusted. The transport layer currently supports UDP and TCP. The overlay layer contains the different P2P overlay protocols, which PeerfactSim.KOM is capable of simulating. The service layer offers additional services, the most important one is monitoring. The application layer hosts P2P applications. Currently, only a file sharing application is implemented [20].

PeerfactSim.KOM simulator as mentioned above provides implementation of numerous P2P protocols like Pastry, CAN, Gnutella, etc. It also offers a user-friendly graphical user interface (Fig. 7).

For every P2P protocol, that shall be simulated, there is an XML-based configuration file, which specifies the used layers, the P2P protocol, and the maximum number of nodes, as well as the maximum simulation time and the classes for data collection.

For gathering data arising from the simulation, the PeerfactSim.KOM simulator offers logging and statistic functionality. The logging is responsible for recording the events during a simulation like nodes joining and failing. The statistic functionality gathers a considerable amount of statistical data from the simulation, such as number of failed messages, sent messages and received messages, as well as data from every participating node, such as number of neighbors, number of leafs, messages sent per second, etc.

Together with the considerable amount of gathered information, the simulator is able to give a visualization of the simulation with rich data in the GUI. Fig. 8 shows a screenshot of the visualization´s main window in case of Pastry-simulation.


PeerfactSim.KOM - Select Launch Configuration

Seed options
[✓ From last run (106354040)]
New seed (-1824606206)
From config (500)
Use custom seed:
0

Variables

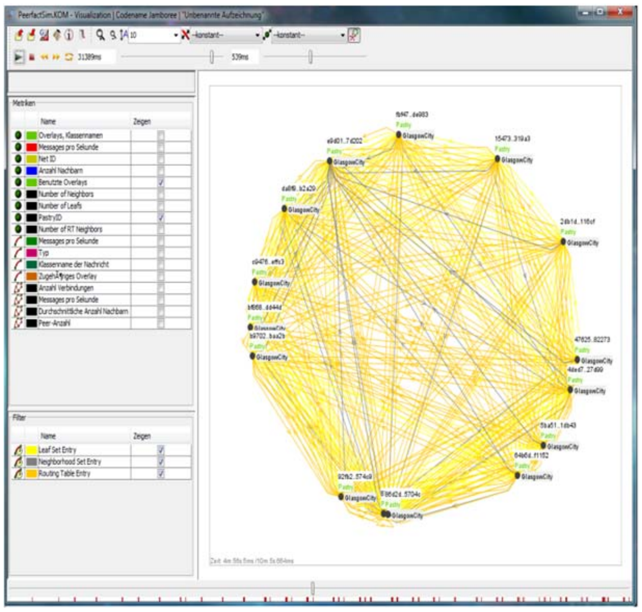
Name	Value
finishTime	120m
actions	config/visualization/pastry-action...
gnpDataFile	data/measured_data.xml
churn	true
NetLayer	Mod

Description

```
PeerfactSim Configurations
  Recently opened
    pastry_30.xml
  Available configs
    aggr
      AChord2-1000.xml
      AGia-1000.xml
      AGnutella-1000.xml
      AKademlia2-1000.xml
    chord.xml
    filesharing2
      FSChord2-1000.xml
      FSGia-1000.xml
      FSGnutella-1000.xml
      FSKademlia2-1000.xml
    Gnutella06.xml
    IDO
      client_server.xml
      mercury.xml
      pSense.xml
      pSense_visualization.xml
      von.xml
    kademlia.xml
    mobile
      MobileFSKademlia2-100.xml
      MobileFSKademlia2-1000.xml
      MobileFSKademlia2-316.xml
    pastry.xml
    skynet
      skynetOnChord.xml
      skynetOnNapster.xml
    visualization
      chord2.xml
      filesharing2-gnutella06.xml
      gnutella04_bitmap.xml
      gnutella04_with_bandwidth.xml
      pastry_30.xml
      pSense_visualization.xml
```

Figure 7. PeerfactSim.KOM ´s Graphical user interface

After the simulation is finished, a window is opened, showing an interface where the simulation can be reviewed in a media player-like fashion. A graphical view displays the nodes and their inter-network connections.


PeerfactSim.KOM - Visualization | Codename Jamboree | "Unbenannte Aufzeichnung"

Metriken

Name	Zeigen
Overlays, Klassennamen	
Messages pro Sekunde	
Net ID	
Anzahl Nachbarn	
Benutzte Overlays	✓
Number of Neighbors	
Number of Leafs	
PastryID	✓
Number of RT Neighbors	
Messages pro Sekunde	
Typ	
Klassenname der Nachricht	
ZugehÃ¶riges Overlay	
Anzahl Verbindungen	
Messages pro Sekunde	
Durchschnittliche Anzahl Nachbarn	
Peer-Anzahl	

Filter

Name	Zeigen
Leaf Set Entry	✓
Neighborhood Set Entry	✓
Routing Table Entry	✓

Figure 8. Visualization´s main window in case of pastry simulation

Various parameters can be viewed and displayed for the nodes and their connections like nodeId, number of neighbors, type of messages, etc. The time variant process can be started, stopped, paused, fast forwarded and rewound. The execution speed can be slowed and accelerated. Another valuable feature of the simulator is the Gnuplot-Export. The previously mentioned parameters can be plotted with the help of Gnuplot and displayed as functions over simulation time. A convenient visualization can be achieved this way.

PeerfactSim.KOM simulator offers the main block for testing and evaluating the functioning of the PeCoCC framework. It can be run in the eclipse environment and extended with more applications and protocols, which present the work of PeCoCC framework. As it is mention in the previous section, the Pastry overlay has been chosen to serve some real time applications like a distributed text editor and a shared calendar. The PeerfactSim.KOM simulator provides an implementation of the Pastry overlay but lacks some more important functions, which are known as applications of pastry itself. These functions are needed to fulfill the requirements of running real time applications on the top of the Pastry overlay. They comprise SCRIBE, the overlay multicasting protocol as well as POST, the co-operative messaging system, which extent the functionality of Pastry algorithm.

B. *Pastry implementation in PeerfactSim.KOM*

As one of DHT algorithms, Pastry is implemented in the package `impl/overlay/dht/pastry` at overlay network layer in PeerfactSim.KOM simulator. The implementation consists of many sub packages that define the components, messages and operations of Pastry. Fig. 9 shows a hierarchy presentation of the implementation of Pastry´s sub packages in PeerfactSim.KOM simulator. Some important classes in these sub packages will be described briefly in the following paragraphs.

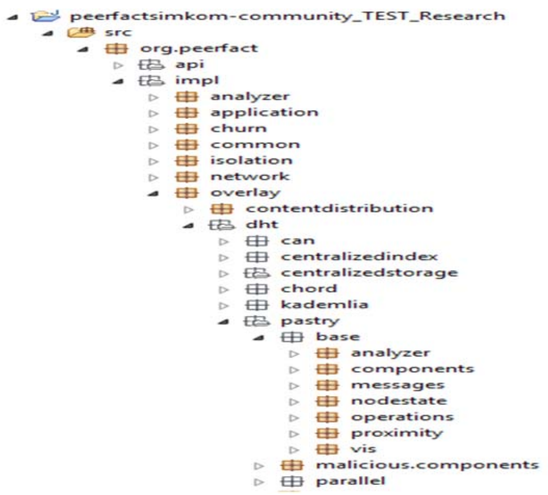

```
◢ peerfactsimkom-community_TEST_Research
  ◢ src
    ◢ org.peerfact
      ▷ api
      ◢ impl
        ▷ analyzer
        ▷ application
        ▷ churn
        ▷ common
        ▷ isolation
        ▷ network
        ◢ overlay
          ▷ contentdistribution
          ◢ dht
            ▷ can
            ▷ centralizedindex
            ▷ centralizedstorage
            ▷ chord
            ▷ kademlia
            ◢ pastry
              ◢ base
                ▷ analyzer
                ▷ components
                ▷ messages
                ▷ nodestate
                ▷ operations
                ▷ proximity
                ▷ vis
              ▷ malicious.components
              ▷ parallel
```

Figure 9. Pastry implementation in PeerfactSim.KOM simulator

Many important components of the Pastry overlay network are declared in the sub package `pastry/base/component`. For example, the class `PastryID` represents the overlay ID used by Pastry. Also the class `PastryNode` is used to represent node of the Pastry overlay and how it manages its leaf and neighbor sets as well as routing table. The method `route(msg, key)` is also declared in this class. The class `PastryMessageHandler` handles incoming messages and dispatches or processes them. The method `send(msg, IP)` is declared in this class.

All the messages of Pastry overlay are declared in the sub `package pastry/base/messages`. For example, the class `LookupMsg` is used to find the responsible node for a given target ID. Also the class `JoinMsg` represents the message sent by a node to join the pastry overlay.

In the sub package `pastry/base/operations` many operations of the pastry overlay are declared; e.g., look up and join operations. The following paragraph presents the primary implementation of Scribe in PeerfactSim.KOM simulator, which is still under development.

C. *SCRIBE*

As it is mentioned in Section III, Scribe is built on the top of Pastry; it heavily relies on its functionality. With the help of Pastry´s methods `route(msg, key)` and `send(msg, IP)`, the four messages types of Scribe can be distributed in the network. These messages are

- `create(credentials, nodeId)`,
- `join(credentials, nodeId)`,
- `leave(credentials, nodeId)` and
- `multicast(credentials, nodeId, message)`.

All Scribe messages use the `route` or `send` method of Pastry, where the key is the according groupID of the message, which will be described later. Scribe also has its two own methods, which instruct the nodes of the network how to handle Scribe messages. These methods are `forward` and the `deliver`.

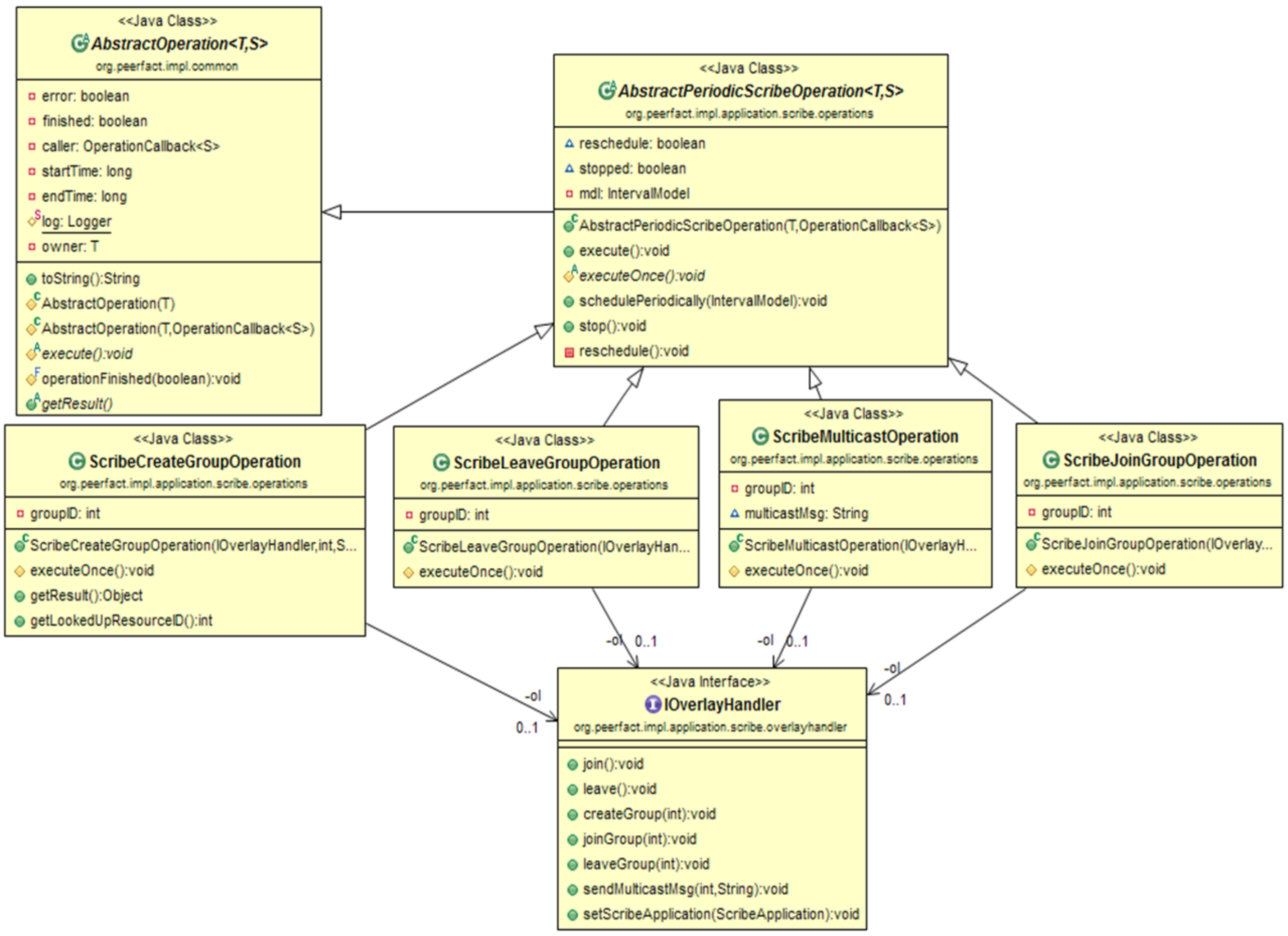
In order to implement application specific behavior of Scribe, changes and extensions of the code have been performed on the application layer and overlay layer of the simulator. On the application layer, the behavior of the application encountering a certain event of the simulation is defined. On the overlay layer, the pastry implementation has been adapted to process and react to Scribe application behavior. Furthermore, a scenario triggering certain application behavior (called actions) has been defined in a file, as well as the simulator configuration in a configuration file.

1) Adaption of the Application Layer: The application has been divided into three functional units ensuring the desired behavior. The core of the Scribe application, which comprises the first functional unit, consists of the two java classes `ScribeApplicationFactory` and `ScribeApplication`. The first class ensures that every participating node of the network runs an instance of the Scribe application. In the second class, the methods `join` and `leave`, which enable the node to join and leave the Scribe application in the overlay network, are defined. Also, all basic methods, which realize the application specific behavior (`createGroup`, `joinGroup`, `leaveGroup`, `sendMulticastMsg`) are declared in `ScribeApplication` class. These methods in turn call specific corresponding operations that form the second functional unit of Scribe implementation.

These operations are

- `ScribeLeaveOperation`,
- `ScribeCreateGroupOperation`,
- `ScribeJoinGroupOperation`,
- `ScribeLeaveGroupOperation`
- `ScribeMulticastOperation`
- `ScribeJoinOperation`

Fig. 10 presents an UML diagram of this part of implementation. Basically, all operations call the methods of functional unit three with the help of the method `executeOne`. The third functional unit adapts the general application specific behavior to the Pastry overlay. The class `PastryHandler` contains all Pastry specific methods, which are called by the method `executeOne` of the according operation: `join`, `leave`, `createGroup`, `joinGroup`, `leaveGroup` and `sendMulticastMsg`. These methods execute the `route` method of Pastry, which contains the according key and Scribe message defined by the user of the application. The information, which key and Scribe message type to use are passed down from the user to the methods of the application layer. With these specifications in the application layer, the simulator is enabled to trigger the behavior of Scribe application according to incoming events specified by the user. In order to process and react on this behavior, the Pastry implementation needed to be adapted on the overlay layer as well.



<<Java Class>>
AbstractOperation<T,S>
org.peerfact.impl.common
error: boolean
finished: boolean
caller: OperationCallback<S>
startTime: long
endTime: long
log: Logger
owner: T
toString():String
AbstractOperation(T)
AbstractOperation(T,OperationCallback<S>)
execute():void
operationFinished(boolean):void
getResult()
<<Java Class>>
AbstractPeriodicScribeOperation<T,S>
org.peerfact.impl.application.scribe.operations
reschedule: boolean
stopped: boolean
mdl: IntervalModel
AbstractPeriodicScribeOperation(T,OperationCallback<S>)
execute():void
executeOnce():void
schedulePeriodically(IntervalModel):void
stop():void
reschedule():void
<<Java Class>>
ScribeCreateGroupOperation
org.peerfact.impl.application.scribe.operations
groupID: int
ScribeCreateGroupOperation(IOverlayHandler,int,S...
executeOnce():void
getResult():Object
getLookedUpResourceID():int
<<Java Class>>
ScribeLeaveGroupOperation
org.peerfact.impl.application.scribe.operations
groupID: int
ScribeLeaveGroupOperation(IOverlayHan...
executeOnce():void
<<Java Class>>
ScribeMulticastOperation
org.peerfact.impl.application.scribe.operations
groupID: int
multicastMsg: String
ScribeMulticastOperation(IOverlayH...
executeOnce():void
<<Java Class>>
ScribeJoinGroupOperation
org.peerfact.impl.application.scribe.operations
groupID: int
ScribeJoinGroupOperation(IOverlay...
executeOnce():void
-ol
0..1
-ol
0..1
-ol
0..1
-ol
0..1
<<Java Interface>>
IOverlayHandler
org.peerfact.impl.application.scribe.overlayhandler
join():void
leave():void
createGroup(int):void
joinGroup(int):void
leaveGroup(int):void
sendMulticastMsg(int,String):void
setScribeApplication(ScribeApplication):void

Figure 10. UML diagram for operation classes of Scribe´s implementation in application layer

2) Adaption of the Overlay Layer: The implementation of Pastry on the overlay layer contains several functional units, of which only two needs to be adapted.

In the first functional unit, the four kinds of Scribe messages, which use the method `route`, need to be defined at the overlay layer implementation of Pastry. The definitions of the messages are realized in the following java classes `SCRIBEcreateGroupMsg`, `SCRIBEjoinMsg`, `SCRIBEleaveMsg`, and `SCRIBEmulticastMsg`. The second functional unit defines the basic components and functionalities of the Pastry algorithm. The Processing of the Scribe messages are defined in the class `PastryMessageHandler`, which handles incoming messages and dispatches or processes them in every Pastry node. The `forward` and `deliver` methods of the original Scribe specification are also implemented there for every Scribe Messages. Fig. 11 presents the UML diagram of a part of Scribe implementation in overlay network.

Furthermore, the class `PastryNode` has been adapted in order to maintain the message groups and children. Hence, the class `PastryMessageHandler` reacts on incoming Scribe messages and calls group management methods of `PastryNode`. Thus, the Pastry implementation of the simulator has been enabled to process and react on incoming Scribe messages.

3) Simulation Scenario: In order to adapt the simulator to the newly implemented application, a configuration file had to be created. In configuration files, every component of the simulator can be defined. For the desired Scribe application, an entry of the Scribe application had to be added to the application component of the simulator, as well as an entry of the Pastry overlay to the overlay component. Furthermore, basic parameters of a simulation scenario can be defined, such as total number of nodes or overall simulation time. Since our implementation aims to create collaborative group up to 20 members in a short-term scenario, the variable size, which presents the number of the nodes, has been set to value 20. Fig. 12 illustrates the variables of the Scribe configuration file.

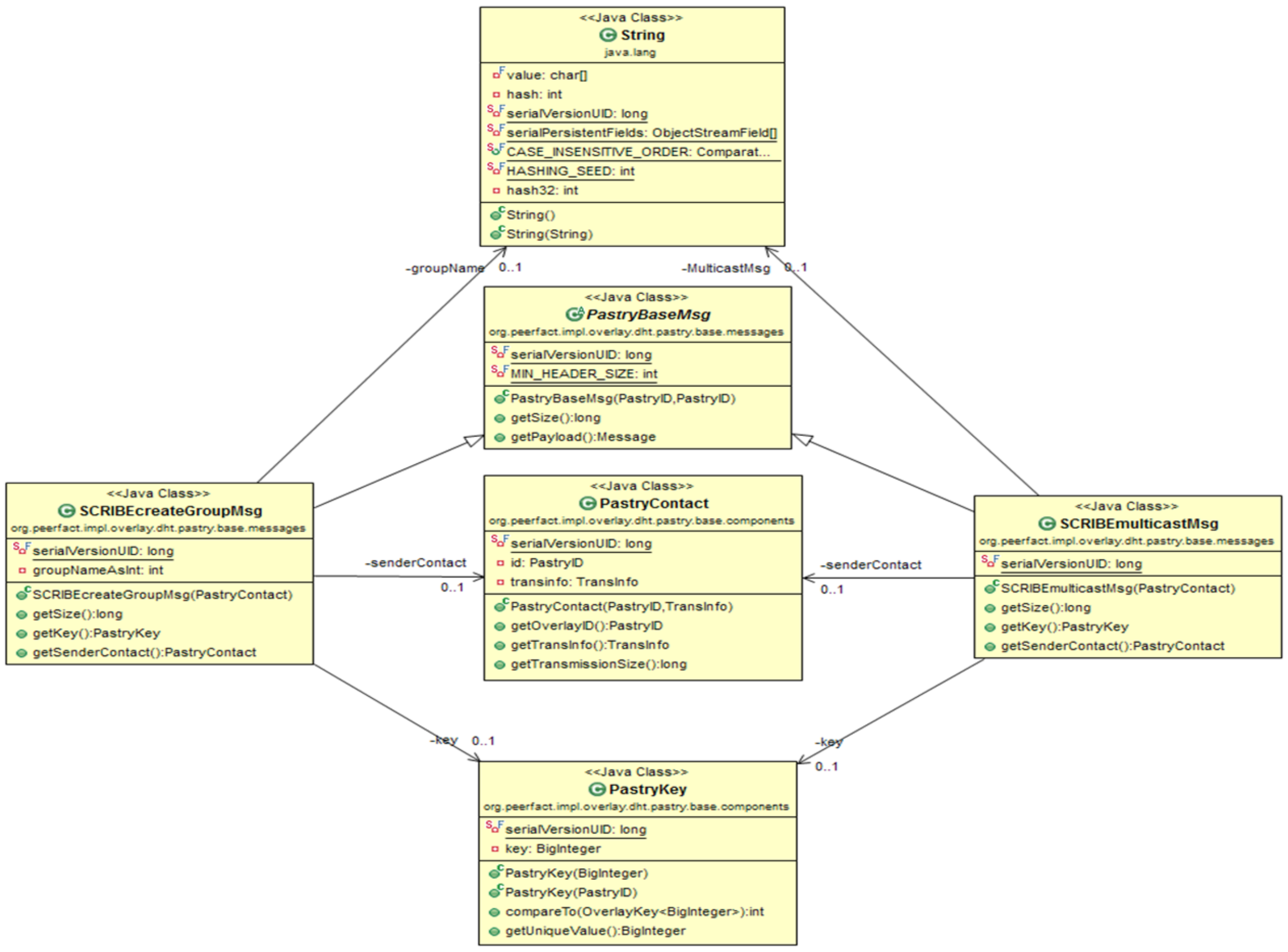
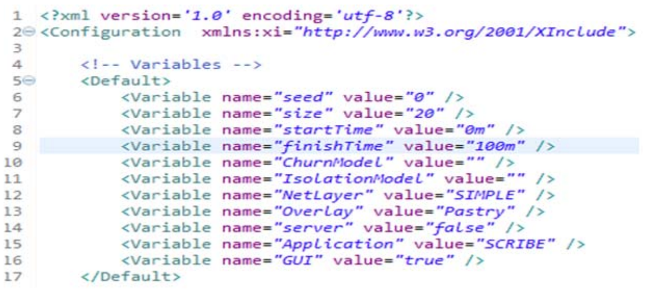


Figure 11. UML diagram for SCRIBEcreateGroupMsg and SCRIBEmulticastMsg classes at overlay layer

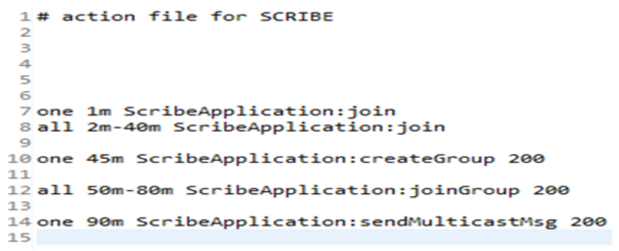
Every simulation scenario comprises several events, which need to be simulated. In the case of Scribe these could be: a node creates a group, another node joins the group, a third node sends a multicast message to the group, etc. In order to trigger these events in the simulation, an action file must be created. In the action file, all events are described by according nodeId, simulation time, application tag, and the events themselves. The simulator then processes the action file and triggers the according behavior at the application-layer by directly calling the methods of the class `ScribeApplication`.



```
<?xml version='1.0' encoding='utf-8'?>
<Configuration  xmlns:xi="http://www.w3.org/2001/XInclude">

    <!-- Variables -->
    <Default>
        <Variable name="seed" value="0" />
        <Variable name="size" value="20" />
        <Variable name="startTime" value="0m" />
        <Variable name="finishTime" value="100m" />
        <Variable name="ChurnModel" value="" />
        <Variable name="IsolationModel" value="" />
        <Variable name="NetLayer" value="SIMPLE" />
        <Variable name="Overlay" value="Pastry" />
        <Variable name="server" value="false" />
        <Variable name="Application" value="SCRIBE" />
        <Variable name="GUI" value="true" />
    </Default>
```

Figure 12. The variables of the Scribe configuration file

To test the implementation of Scribe, which is still in progress and need to be analyzed with better monitoring tools by the simulator, the following simple scenario has been accomplished. In the Fig. 13, the action file, which contains the simulation scenario of Scribe application, can be shown. In the scenario, there is a node called *one* as well as nineteen nodes called *world* summarized to a group called *all*. In the first minute of the simulation, node *one* joins the Scribe application. Between the minutes 2 and 40, the nineteen remaining nodes join the Scribe application. At minute 45, node *one* creates a group with groupID 200dec, which is randomly generated. Between the minutes 50 and 80, all remaining nodes join the created group. Finally, the node *one* sends a multicast message to the created group in the minute 90.



```
# action file for SCRIBE





one 1m ScribeApplication:join
all 2m-40m ScribeApplication:join

one 45m ScribeApplication:createGroup 200

all 50m-80m ScribeApplication:joinGroup 200

one 90m ScribeApplication:sendMulticastMsg 200

```

Figure 13. Scribe Simulation Scenario

To be able to investigate the functions of the new implemented Scribe application, PeerfactSim.KOM simulator provides the possibility to collect the output data and write them into a file that can be used for plotting with the tool `Gnuplot`. The Fig. 14 presents a line graph showing the number of initiated operations in the simulation scenario according to the time of simulation.

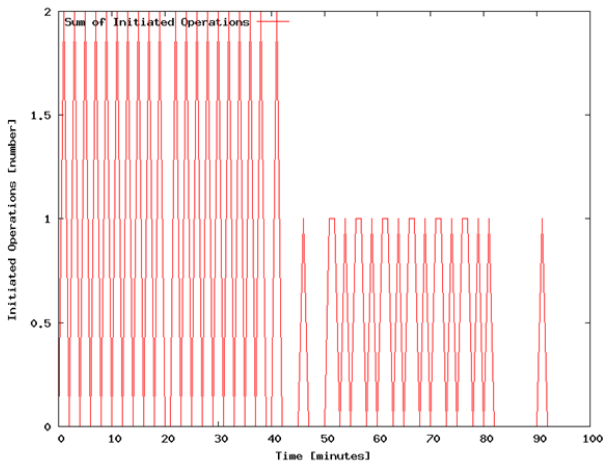


Figure 14. The number of initiated operations of Scribe application in the time of simulation scenario

After all nodes joined the Scribe application at the minute 40, one operation is executed in the minute 45, which is supposed to be the operation of creating a multicasting group. The operations of joining the multicasting group by the host group *all*, which consists of nineteen nodes, have taken place between the minutes 50 and 80 as it is explained in the action file. Whereas, the last prong at the minute 90 embodied the sending of multicast message.

On the other hand, the Fig. 15 shows the number of messages sent by the peers themselves to maintain the overlay and to join the Scribe application during the time of the simulation scenario.

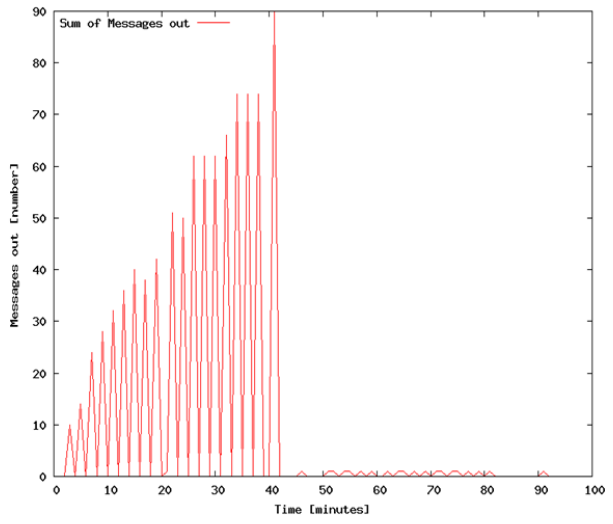


Figure 15. The number of sent messages in the time of simulation scenario

As mentioned in the action file of Scribe, all nodes have to be joined the Scribe application before the minute 40. That means the messages traffic during this time will be more active than after minute 40, which is illustrated in the Fig. 15.

The first results of this implementation showed some needs for more monitoring and analyzing functions of Peerfactsim.KOM simulator like the existence of a visualization analyzer and multicast message analyzer. Furthermore, the implementation has to be improved by extending the multicast message and testing it with many created groups.

VI. Conclusion and Future Work

In this paper, an overview of the current P2P collaborative environments and their use case has been presented. The P2P technologies have been briefly highlighted. We have also introduced our PeCoCC framework to allow computer-supported collaborative learning based on pure P2P networks that provide fully distributed and server-independent P2P communications and services, which increase the availability of these services and solve the problem of single point of failure present in server-based systems. The PeCoCC framework is currently in implementation phase. It supports interoperability and is being implemented in Java using the integrated development environment eclipse. To evaluate the performance of the framework, a P2P simulator named PeerfactSim.KOM is used. This simulator has a similar layered architecture like the PeCoCC framework and supports many forms of messages to communicate among the layers in the host. It is also implemented in Java and it offers a user-friendly graphical user interface. This simulator lacks some important protocols and components that will be needed to evaluate the PeCoCC framework. One of these protocols is the Scribe overlay multicasting protocol, which is also highlighted in this paper. The main components of the Scribe multicasting protocol have been implemented in the PeerfactSim.KOM simulator environment.

In the future, more extensive simulations of Scribe will be performed, i.e., simulations with a large amount of created groups, adding churn to the network, extending the multicast message to be able to contain more complicated content and investigating the network load and message routing latency by some collaborative scenarios. Furthermore, the structure of the PeCoCC framework, i.e., management and control module, overlay selection module and the PeCoCC collaborative messaging scheme have to be refined.

References

[1] M. Hasan and J. Seitz, "Peer-to-peer communication for computer-supported collaborative learning. The PeCoCC framework," In the Sixth International Conference on Mobile, Hybrid, and On-line Learning (eLmL2014) IARIA, Mar. 2014, pp. 25-29, ISSN: 2308-4367, ISBN: 978-1-61208-328-5

[2] G. Woodill, "Computer supported collaborative learning in education and training: Tools and Technologies," Phil. Trans. Brandon Hall Research. San Jose. CA. USA, 2008.

[3] M. Hauswirth, I. Podnar, and S. Decker, "On P2P collaboration infrastructures," Proc. 14th IEEE International Workshop on Enabling Technologies: Infrastructure for collaborative Enterprise, IEEE Press, June 2005, pp. 66-71, DOI: 10.1109/WETICE.2005.47.

[4] J. Vassileva, "Harnessing P2P power in the classroom," Proc. 7th International Conference, ITS, Aug. 2004, pp. 305-314, doi:10.1007/978-3-540-30139-4_29.

[5] M. Miettinen and J. Kurhila, "EDUCOSM – Personalized writable web for learning communities," Proc. Information Technology: Coding and Computing, ITCC, Apr. 2003, pp. 37-42, DOI:10.1109/ITCC.2003.1197496.

[6] C. Qu and W. Nejdl, "Interacting the edutella/JXTA peer-to-peer network with web services," Proc. IEEE Symp. Application and the Internet, 2004, pp. 67-73, doi:10.1109/SAIT.2004.1266100.

[7] T. Smith, "Sharepoint 2013 user´s guide: learning microsoft´s business collaboration platform," 4th ed, Apress, 2013, ISBN: 978-1-4302-4833-0.

[8] E. K. Lua, J. Crowcroft, M. Pias, R. Sharma, and S. Lim, "A survey and comparison of peer-to-peer overlay network schemes," Communications surveys & tutorials, IEEE, vol. 7, 2005, pp. 72-93, DOI:10.1109/COMST.2005.1610546.

[9] A. Rowstron and P. Druschel, "Pastry: scalable, decentralized object location and routing for larg-scale peer-to-peer systems," In 18th IFIP/ACM International Conference on Distributed Systems Platforms, Nov. 2001, pp. 329-350, ISBN: 3-540-42800-3.

[10] M. Castro, P. Druschel, A. Ganesh, A. Rowstron, and D. S. Wallach, "Secure routing for structure peer-to-peer overlay networks," Proc. 5th Symposium on Operating Systems Design and Implementation, ACM Sigops Operation Systems Review, 2002, pp. 299-314, DOI: 10.1145/844128.844156.

[11] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron, "Scribe: A large-scale and decentralized application-level multicast infrastructure," IEEE Journal on Selected Areas in communications, vol. 20, pp. 100-110, Oct. 2002, DOI: 10.1109/JSAC.2002.803069.

[12] J. Nogueira, "A large-scale and decentralised application-level multicast infrastructure,". [Online]. Available from: http://www.gsd.inesc-id.pt/~ler/docencia/tm0607/slides/Scribe-JoaoNogueira.pdf, 2014.12.01.

[13] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shanker, "A scalable content-addressable network," Proc. conference on Applications, technologies, architectures, and protocols for computer communications, SIGCOMM 01, 2001, pp. 161-172, DOI: 10.1145/383059.383072.

[14] S. Ertel, "Unstructured P2P networks by example: Gnutella0.4, Gnutella0.6,". [Online]. Available from: http://ra.crema.unimi.it/turing/materiale/admin/corsi/sistemi/lezioni/m3/m3_u2_def/ceravolo_file2.pdf. 2014.12.01

[15] D. Boukhelef and H. Kitagawa, "Efficient load balancing techniques for self-organizing content addressable networks," Journal of Networks, vol. 5, Mar. 2010, pp. 321-334, doi:10.4304/jnw.5.3.321-33

[16] M.F.M. Firdhous, "Multicasting over overlay networks – a critical review," International Journal of Advanced Computer Science and Applications, vol. 2, pp. 54-61, Mar. 2011, ISSN 2156-5570.

[17] S. Ratnasamy, M. Handley, R.M. Karp, and S. Shenker, "Application-level multicast using content-addressable network," In the third International COST264 Workshop on Networked Group Communication, Oct. 2001, pp. 14-29, ISBN:3-540-42824-0

[18] “The FreePastry tutorial,”. [Online]. Available from: https://trac.freepastry.org/wiki/FreePastryTutoriaal, 2014.12.01

[19] A. Montresor and M. Jelasity, “PeerSim: a scalable P2P simulator,” Proc. IEEE Ninth International Conference on Peer-to-Peer Computing, 2009, pp. 99-100, DOI: 10.1109/P2P.2009.5284506.

[20] “PeerfactSim.KOM documentation,”. [Online]. Available from: https://sites.google.com/site/peerfactsimkom/documentation, 2014.12.01.

PonderFlow: A New Policy Specification Language to SDN OpenFlow-based Networks

Bruno Lopes Alcantara Batista   Marcial Porto Fernandez
Universidade Estadual do Ceará (UECE)
Av Silas Munguba 1700 - Fortaleza/CE - Brazil
{bruno,marcial}@larces.uece.br

Abstract—**The SDN/OpenFlow architecture is a proposal from the Clean Slate initiative to define a new Internet architecture where network devices are simple, and the control plane and management are performed on a centralized controller, called Openflow controller. Each Openflow controller provides an Application Programming Interface (API) that allows a researcher or a network administrator to define the desired treatment to each flow inside controller. However, each Openflow controller has its own standard API, requiring users to define the behavior of each flow in a programming or scripting language. It also makes difficult for the migration from one controller to another one, due to the different APIs. This paper proposes the PonderFlow, an extension of Ponder language to OpenFlow network policy specification. The PonderFlow extends the original Ponder specification language allowing to define an Openflow flow rule abstractly, independent of Openflow controller used. Some examples of OpenFlow policy will be evaluated showing its syntax and the grammar validation.**

Keywords–Openflow; OpenFlow Controller; Policy-based Network Management; Policy Definition Language

I. Introduction

This paper is an extended version of the paper presented in The Thirteenth International Conference on Networks (ICN 2014) [1]. Comparing to the original paper, this one shows more description examples and evaluates the parser implementation to validate the proposal.

The Software Defined Network (SDN) architecture with OpenFlow protocol is a proposal of the Clean Slate initiative to define an open protocol to sets up forward tables in switches [2]. It is the basis of the SDN architecture, where the network can be modified dynamically by the user, and the control-plane are decoupled from the data-plane. The OpenFlow proposal tries to use the most basic abstraction layer of the switch to achieve better performance. The OpenFlow protocol can set a condition-action tuple on switches like forward, filter and also, count packets from a specific flow that match a condition.

The network management is performed by the OpenFlow Controller maintaining the switches simple, only with the packet forwarding function. This architecture provides several benefits: (1) OpenFlow controller can manage all flow decisions reducing the switch complexity; (2) A central controller can see all networks and flows, giving global and optimal management of network provisioning; (3) OpenFlow switches are relatively simple and reliable, since forward decisions are defined by a controller, rather than by a switch firmware. However, as the number of switches increases in a computer network and it becomes more complex to manage the switches flows, it is necessary to use a tool to help the network administrator to manage the flows in order to modify dynamically the system behavior.

A policy-based tool can reduce the complexity inherent to this kind of problem. It is a way to manage a large network environment, where the behavior of the network assets may change over time.

Policy-Based Network Manager (PBNM) is the technology that provides the tools for automated network using policies to describe environment behavior abstractly. The PBNM can help network administrators to manage OpenFlow networks simply defining policies, where a policy is a set of rules to govern all the system.

This paper presents the PonderFlow, an extension of Ponder policy specification language. Ponder is a declarative, object-oriented language for specifying management and security policy proposed by Damianou et al. [3]. The PonderFlow provides the necessary resources to define or remove flows, grant privileges to a user, add or remove flows (authorization policy) and force a user or system to execute an action before a particular event (obligation policy).

The rest of the paper is structured as follows. In Section II, we present some related work about OpenFlow policy specification languages. Section III introduces the OpenFlow, the Policy-Based Openflow Network Manager (PBONM) architecture and introduces the Ponder specification language. In Section IV, we present the PonderFlow language, and its respective grammar and validation. In Section VI, we conclude the paper and present some future works.

II. Related Work

Foster et al. [4] designed and implemented the Frenetic, a set of Python's libraries for network programming to provide several high-level features for OpenFlow/NOX [5] programming issues. Frenetic is based on functional reactive programming, a paradigm in which programs manipulate streams of values, delivering the need to write event-driven programs leading a unified architecture where programs "see every packet" rather than processing traffic indirectly by manipulating switch-level rules. However, the network administrator needs to use a programming language, Python [6] in this case, to define the behavior of OpenFlow network.

Mattos et al. [7] propose an OpenFlow Management Infrastructure (OMNI) for controlling and managing OpenFlow networks and also for allowing the development of autonomous applications for these networks. OMNI provides a web interface with set of tools to manage and control the network, and the network administrators interact through this interface. The outputs of all OMNI applications are eXtensible Markup Language (XML), simplifying the data interpretation by other applications, agents or human operators. However, the network administrator needs to use a programming language to call any OMNI function using a web Application Programming

Interface (API) or access the web interface and proceed manually.

Voellmy et al. proposed Procera [8], a controller architecture and high-level network control language that allow to express policies in the OpenFlow controllers. Procera applies the principles of functional reactive programming to provide an expressive, declarative and extensible language. Users can extend the language by adding new constructors.

The PonderFlow has similarities with Procera and Frenetic, but our main goal is to create a policy specification language decoupled from the conventional programming languages, and also, regardless of the OpenFlow controller used. The PonderFlow language is an extension of Ponder language and can be easily ported to another OpenFlow controller. As Ponder is a well-known policy language, the validations is not necessary. In this work, we used the Java language to implement the parser and lexical analyses in Floodlight OpenFlow controller [9]. We want to achieve a level of independence from the programming language and of the OpenFlow controllers. This paper presents the PonderFlow, an extensible, declarative language for policy's definition in an OpenFlow network.

III. OpenFlow Policy Architecture

In this section, we introduce SDN architecture and the Policy-based network management concepts. We also show the application of Policy-based management architecture in OpenFlow environment.

A. *OpenFlow*

The SDN architecture has several components: the OpenFlow controller, one or many OpenFlow devices (switch), and the OpenFlow protocol. This approach considers a centralized controller that configures all devices. Devices should be kept simple in order to reach better forward performance and leave the network control to the controller.

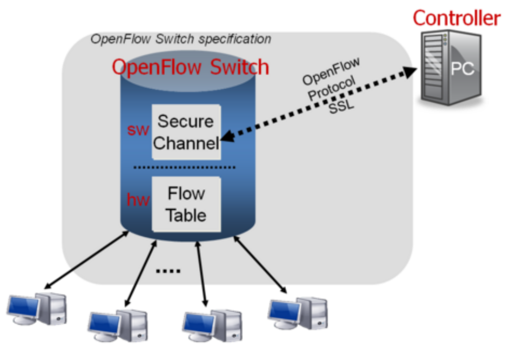


Figure 1. The OpenFlow architecture [2]

The OpenFlow Controller is the centralized controller of an OpenFlow network. It sets up all OpenFlow devices, maintains topology information, and monitors the overall status of entire network. The device is any capable OpenFlow device on a network such as a switch, router or access point. Each device maintains a Flow Table that indicates the processing applied to any packet of a certain flow. There are several OpenFlow controllers available, e.g., NOX [5], FloodLight [9], Beacon [10], POX [11], and Trema [12].

The OpenFlow Protocol works as an interface between the controller and the OpenFlow devices setting up the Flow Table. The protocol should use a secure channel based on Transport Layer Security (TLS). The controller updates the *Flow Table* by adding and removing Flow Entries using the OpenFlow Protocol. The Flow Table is a database that contains Flow Entries associated with actions to command the switch to apply some actions on a certain flow. Some possible actions are: forward, drop, and encapsulate.

The Openflow Controller presents two behaviors: reactive and proactive. In the **Reactive** approach, the first packet of flow received by switch triggers the controller to insert flow entries in each OpenFlow switch of network. This approach presents the most efficient use of existing flow table memory, but every new flow incurs in a small additional setup time. Finally, with hard dependency of the controller, if a switch lost the connection, it has limited utility.

In the **Proactive** approach, the controller pre-populates flow table in each switch. This approach has zero additional flow setup time because the forward rule is defined. Now, if the switch loss the connection with controller it does not disrupt traffic. However, the network operation requires a hard management, e.g., requires to aggregate (wildcard) rules to cover all routes.

Each device has a Flow Table with flow entries as shown in Figure 2. A Flow Entry has three parts: rule match fields, an action and statistics fields and byte counters. The *rule match fields* is used to define the match condition to a specific flow. *action* defines the action to be applied to an exact flow, and *statistical fields* are used to count the rule occurrence for management purposes.

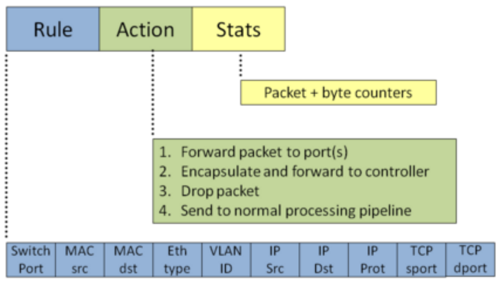


Figure 2. The OpenFlow Flow Entry [13]

When a packet arrives to the OpenFlow Switch, it is matched against *flow entries* in the *flow table*, and the action will be triggered if the header field is matched and then updates the counter.

If the packet does not match any entry in the *flow table*, the packet will be sent to the controller over a secure channel. Packets are matched against all *flow entries* based on a prioritization, where each *flow entry* on *flow table* has a priority associated. Higher numbers have higher priorities.

The OpenFlow Protocol uses the TCP protocol and port 6633. Optionally, the communication can use a secure channel based on TLS.

The OpenFlow Protocol supports three types of messages [13]:

1) Controller-to-Switch Messages: These messages are sent only by the controller to the switches, and they perform the functions of switch configuration, exchange information on the switch capabilities and also manage the Flow Table.

2) Symmetric Messages: These messages are sent in both directions reporting on switch-controller connection problems.

3) Asynchronous Messages: These messages are sent by the switch to the controller to announce changes in the network and switch state.

All packets received by the switch are compared against the Flow Table. If the packet matches any Flow Entry, the action for that entry is performed on the packet, e.g., forward a packet to a specified port. If there is no match, the packet is forwarded to the controller that is responsible for determining how to handle packets without valid Flow Entries [13].

It is important to note that when the OpenFlow switch receives a packet to a nonexistent destination in the Flow Table, it requires an interaction with the controller to define the treatment of this new flow. At least, the switch will need to send a message to the controller with regards to the unknown packet received (message Packet-In). The controller needs to configure all switches along the path from source to destination (message Modify-State). In a network with N switches, we can estimate the need of $6 \times (N + 1)$ messages for each new flow, considering the start and end of TCP connection, the Packet-in and Modify Flow Entry Messages [13]. If the path is already pre-defined (there is an entry in Flow Table), this procedure is not necessary, reducing the amount of messages exchanged through the network and reducing the processing at the controller.

Furthermore, the maintenance of old Flow Entries in the switch Flow Tables gives waste of fast Ternary Content-Addressable Memory (TCAM) memory. Therefore, it is required to remove unused flows using a time-out mechanism. If a flow previously excluded by time-out restarts, it is required to reconfigure all switches on the end-to-end path.

An OpenFlow device is basically an Ethernet switch with OpenFlow protocol. However, there are different implementation approaches: OpenFlow-enable switch and OpenFlow-compliant switch.

The OpenFlow-enable switch uses off-the-shelf hardware, i.e., traditional switches with OpenFlow protocol that translate the rule according to the hardware chipset implementation. The OpenFlow enable-switch re-use existing TCAM, that in a conventional switch has no more than only few thousands of entries for IP routing and MAC table. Considering that we need at least one TCAM entry per flow, in a current hardware would be not enough for production environments. The Broadcom chipset switches based on Indigo Firmware [14], e.g., Netgear 73xxSO, Pronto Switch and many other, are an example of this approach.

The OpenFlow-compliant switch uses a specific network chipset, designed to provide better performance to OpenFlow devices. OpenFlow philosophy relies on matching packets against multiple tables in the forwarding pipeline, where the output of one pipeline stage being able to modify the contents of the table of next stage. Some examples are devices based on the EZChip NP-4 Network Processor [15] and Intel FM-6000 [16]. But, nowadays, there are few commercial OpenFlow-compliant switches; one example is the NoviFlow Switch 1.1 [17].

B. Policy Based Network Management

In traditional network management, policies are hard coded and require manual intervention o be modified. The costs of configuration the network results in a manpower intensive task, and it can result a significant portion of network operations because [18]:

- There are many network elements to be configured;
- Network problems require manual intervention;
- Dynamic user demands or network conditions require repeated reconfiguration;
- Manually maintaining consistency and coherence across and between systems is error prone.

PBNM has emerged as a promising paradigm for network operation and management. PBNM has the advantage of being able to change dynamically the behavior of a managed system according to the changing context requirements without having to modify the implementation of managed system [19].

The general PBNM can be considered an adaptation of the IETF policy framework to apply to the area of network provisioning and configuration. The IETF/DMTF policy framework is shown in Figure 3 and consist of four elements:

- *Policy Management Tool* (PMT): Graphical tool to define which policies will be applied in the network.
- *Policy Repository* (PR): It is used for the storage of policies, and it is typically a relational database or a directory.
- *Policy Decision Point* (PDP): It parses the policy, checks the authorization and validity before communicating them to the PEP.
- *Policy Enforcement Point* (PEP): It can apply and execute the different policies into network devices.

With the PBNM, we can simplify the management process through of centralization and business-logic abstractions [19].

Centralization refers to the process of defining all the devices provisioning and configuration at a single point (the PMT) rather than provisioning and configuring each device itself. The benefits of centralization in reducing manual tedium can easily be seen. In a network of 500 machines requires from 10 to 15 minutes of configuration per machine, the network administrator would need to work around 10 a 15 days (on a journey of eight hours of daily work).

Business-level abstractions make the job of the policy administrator simpler by defining the policies in terms of a language closer to the business needs of an organization rather than in terms of the specific technology need to deploy it. The network administrator needs not to be very conversant with the details of the technology that supports the desired need.

C. Policies

Policies [20] are one aspect of information, which influences the behavior of objects within the system. They are often used as a means of implementing flexible and adaptive systems for management of Internet services, distributed systems, and security systems [21].

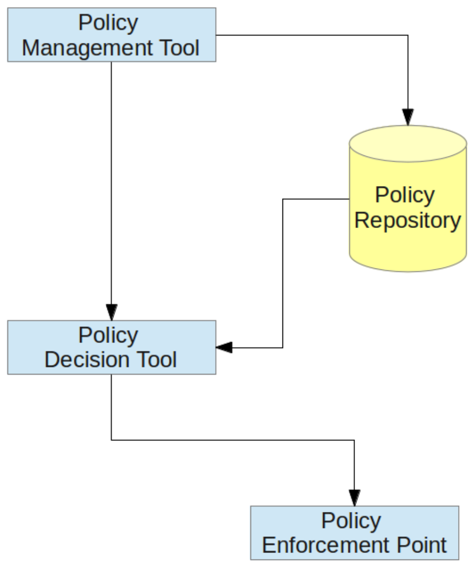


Figure 3. The IETF/DTMF policy framework [19]

Policies are classified into two categories:

- **Authorization Policies**: Define what a manager *is permitted or not permitted to do*. They limit the information made available to managers and the operations they are permitted to perform on managed objects.
- **Obligation Policies**: Define what a manager *must or must not* do and hence guide the decision-making process.

Authorization policies are specified to protect target objects and are usually implemented using security mechanisms when subjects cannot be trusted to enforce them. Obligation policies are event-triggered condition-action rules that can be used to define adaptable management actions [21].

Large-scale systems may contain millions of users and resources, and it is not practical to specify policies relating to individual's entities. It instead must be possible to specify policies relating groups of entities and also to nested groups such as sections within departments, within sites in different countries in an international organization. Domains can be used for this case.

A **Domain** is a collection of managed objects, which have been explicitly grouped together, based on geographical boundaries, object type, responsibility and authority or for covenience of human managers, for the purposes of management [20] [21].

More specifically a domain is a managed object which maintains a list of references to its member managed objects. If a domain holds a reference to an object, the object is said to be a *direct member* of the domain, and the domain are said to be its *parent* [20].

D. Policy Conflicts

In large distributed systems, there will be multiple human administrators specifying policies. Policy conflicts can arise due to omissions, errors or conflicting requirements of the administrators specifying the policies [22].

For example, an obligation policy may define an activity a manage must perform but there is no authorization policy to permit the manager to perform the activity. Obvious conflicts occur if both a positive and negative authorization or obligation policy with the same subjects, targets and actions.

The problem of detecting conflicts is extremely difficult. Analysis of the policy objects without any knowledge of the application or activities may detect positive-negative conflicts of modalities and conflicts between obligation and authorization policies, and so it may be possible to automate this [20].

E. Policy-Based Openflow Network Manager

The behavior of an OpenFlow network is defined by flow table entries of the devices (e.g., switch) comprising the network. These entries determine the action to be taken by the device, which may authorize the entry of a package in the device so that, it can be forwarded to another device or host or deny the packet in the device. However, some questions arise naturally about: (1) How to create or manage OpenFlow network with controllers currently present? (2) How to delegate or revoke network permissions to a particular user? (3) How to manage the switches flows as the number of hosts and switches increases?

Policy-Based Network Manager (PBNM) has emerged as a promising paradigm for network operation and management, and has the advantage to dynamically change the behavior of a managed system according to the context requirements without the need to modify the implementation of managed system [19]. The general PBNM can be considered an adaptation of the Internet Engineering Task Force (IETF) policy framework to apply to the area of network provisioning and configuration.

With PBNM the management network process can be simplified through of centralization and business-logic abstractions [19]. Centralization refers to the process of configuring all devices in a single-point (Policy Management Tool (PMT)) instead of reconfiguring the device individually.

In a previous work [23], we propose to use the PBNM concepts in OpenFlow networks. PBONM was proposed, a framework based on the IETF policy framework. Ponder language was chosen as the standard policy specification language in the PBONM. The PBONM is depicted on Figure 4. The architecture is divided in the following layers:

Policy Management Tool (PMT): it is a software layer that manages the network users, switches and OpenFlow layers providing the User Interface to enable these features. The Ponder is used to specify the policies in this layer. The Policy Repository (database) will store the policies and other information about of the network.

Policy Decision Point (PDP): it is responsible to interpreting the policies stored in the repository, checks the users' authorization (if the user has permission to add or remove a flow in specific switch), check policy conflicts on database and release the policies to Policy Enforcement Point.

Policy Enforcement Point (PEP): it is responsible to execute the configuration of OpenFlow controller. When the policies are interpreted, OpenFlow flows are generated and forwarded to the OpenFlow controller. So, the OpenFlow controller can enforce these flows on the network.

OpenFlow Network Devices: they are OpenFlow switches controlled by an OpenFlow controller and configured by PEP.

The PBONM is depicted on Figure 4. The architecture is divided in the following layers:

- *PMT*: it is a software layer to manage the network users, switches and OpenFlow layers providing the User Interface to enable these features. The Ponder is used to specify the policies in this layer. The database (LDAP or RDBMS) will store the policies, and other informations needed the network.
- *Policy Decision Point(PDP)*: it is responsible to interpreting the policies stored in the repository, checks the users' authorization (if the user has permission to add or remove a flow in specific switch), check policy conflicts on database and release the policies to Policy Enforcement Point.
- *Policy Enforcement Point(PEP)*: it is responsible to execute the configuration of OpenFlow controller. When the policies are interpreted, OpenFlow flows are generated and forwarded to the OpenFlow controller. So the OpenFlow controller can enforce these flows on the network.
- *Network Components(NC)*: t are switches and routers subordinate to OpenFlow controller. These network components are configured by flows sent by PEP.

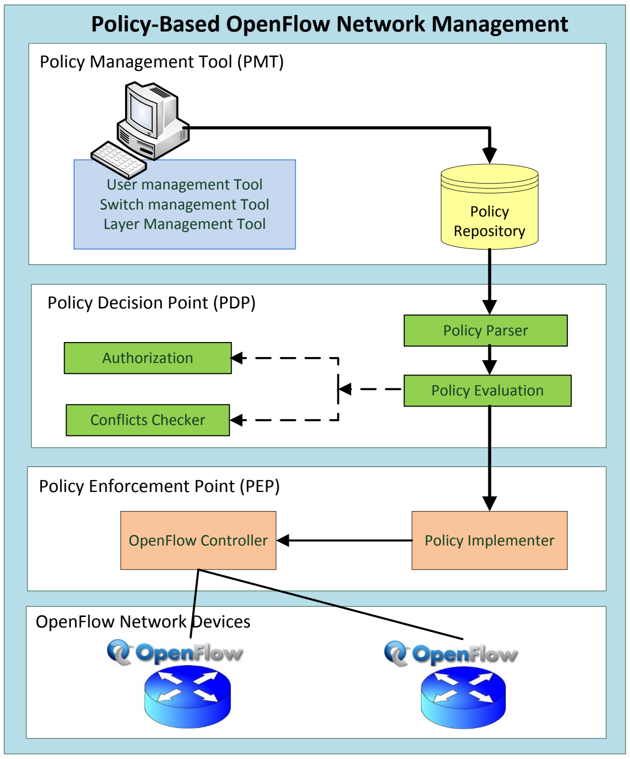


Figure 4. The Policy-Based OpenFlow Network Manager architecture

Thus, the network administrator can specify network flows and the users' permission through of a graphical tool using a policy specification language. These policies will be translated to OpenFlow controller API calls and will be applied to the network devices.

F. Ponder: Policy Specification Language

Ponder is a declarative, object-oriented language for specifying security and management policy for distributed object systems proposed by Damianou et al. [3]. The language is flexible, expressive and extensible to cover the wide range of requirements implied by the current distributed systems requirement and allows for the specification of security policies (role-based access control) and management policies (management obligations) [20].

There are four building blocks supported on Ponder, which are: (1) *authorizations*: what activities the subject can perform on the set of target objects; (2) *obligations*: what activities a manager or agent must perform on target objects; (3) *refrains*: what actions a subject must not execute on target objects; (4) *delegation*: granting privileges to grantees.

However, the Ponder language does not support the network flows abstraction. In contrast, OpenFlow architecture works over the network flows concept. To use Ponder in PBONM, an extension to the language is needed, to support the requirement inherent in the new environment. Thus, a network administrator can define flows in a network switch OpenFlow clearly and concisely.

The advantage of using a policy language is to permit the network administrator only needs to think in an abstract form, how the OpenFlow network should work, without worrying about the implementation details of a specific controller. Unlike other flow language's definition, that requires the administrator to use a programming language [4], [7], [8].

Ponder2 is a re-design of Ponder language and toolkit, maintaining the concepts and the basic constructs [24]. In contrast to the original Ponder, which was designed for general network and systems management; Ponder2 was designed as an extensible framework to configure more complex services. It uses the PonderTalk, a high-level configuration and control language, and it permits user-extensible Java objects. In our proposal, we prefer to use the original Ponder language because the new functionality of Ponder2 is not necessary. We believe that the concise description of Ponder is easier for a network administrator, unlike the more extensible and complex PonderTalk description.

IV. PONDERFLOW: OPENFLOW POLICY SPECIFICATION LANGUAGE

Ponder is the policy language used to manage security policies and access control. However, the Ponder language is too vague to cover all types of manageable environments [25]. PonderFlow is a policy definition language for OpenFlow networks where your main objective is to specify flows transparently, independent of OpenFlow controller used in the network. The PonderFlow extends the Ponder language [3] to suit the flow definition paradigm of OpenFlow environment.

Some of the Ponder's building blocks were kept and others were not used in favor of simplicity. Nevertheless, even keeping some building blocks from the original Ponder language;

the philosophy behind these blocks was changed to suit the paradigm of OpenFlow networks. Furthermore, it was added a way to specify flows through policies, making PonderFlow a declarative scripting language. In this way, the new keyword **flow** is included to specify the flow's characteristics. In the following subsections, the building blocks will be explained, and we will show some examples to manage network flows.

ANTLR framework [26] was used to generate the lexical analyzer and parser grammar in the Java programming language, as well as to generate the images of Abstract Syntax Tree (AST) tree of the building blocks defined in the PonderFlow.

A. Authorization Policies

The authorization policies define what the members within a group (subject) may or may not do in the target objects. Essentially, these policies define the level of access the users possess to use an OpenFlow switches network.

A positive authorization policy defines the actions that subjects are permitted to do on target objects. A negative authorization policy specifies the actions that subjects are not allowed to do on target objects.

This building block is very similar to the original language Ponder, but the focus of this building block in PonderFlow context is in the access by the users in the switches that comprise the OpenFlow network and OpenFlow controller itself.

Listing 1. PonderFlow Authorization Policy Sintax

```
inst ( auth+ | auth− ) policyName {
   subject [<type_def>]  domain−scope−expression;
   target  [<type_def>]  domain−scope−expression;
   [ flow  [<type_def>]  flow−expression; ]
   action          action−list;
   [ when constraint−expression |
      constraint−flow−expression ];
}
```

The syntax of an authorization policy is shown in Listing 1. Everything in bold is language keywords. Choices are enclosed within round brackets **( )** separated by **|**. Names and variables are represented within **< >**. Optional elements are specified with square brackets **[ ]**. The policy body is specified between braces **{ }**.

Constraints are optional in authorization policies and can be specified to limit applicability of policies based on time or attribute values to the objects on which the policy refers.

The elements of an authorization policy can be specified in any order, and the policy name must begin with a letter and contain letters, numbers and underscore in the rest of your name.

The specification of the subject and target may be optionally specified using an Uniform Resource Identifier (URI) to represent the domain of the subject or of the target. Moreover, we can specify the subject type or the target type in the policy definition.

Listing 2. Positive authorization policy example

```
inst auth+ switchPolicyOps {
   subject <User>     /NetworkAdmin;
   target  <OFSwitch>  /Nregion/switches;
   action addFlow(), removeFlow(), enable(), disable();
}
```

Listing 2 shows an example of a positive authorization policy allowing all network administrators to perform the actions of adding flows, remove flows, enable and disable all switches in Nregion. Note, this policy is applied to any flow, and it is similar to conventional Ponder authorization policy. In Figure 5, we show the AST tree of a positive authorization policy from Listing 2.

A snippet of ANTLR grammar defined for PonderFlow authorization policies is described in Listing 3.

Listing 3. Authorization policy grammar

```
auth_policy:
       INST ( AUTH_POSITIVE | AUTH_NEGATIVE )
         policy_name
         OPEN_BODY
           auth_policy_options*
         CLOSE_BODY
       | INST ( AUTH_POSITIVE | AUTH_NEGATIVE )
         policy_name SET policy_name
           OPEN_PARENTESIS
             variable ( COMMA variable)*
           CLOSE_PARENTESIS;
```

In Figure 5, we have the AST tree generated by ANTLR to move the policy of positive authorization of Listing 2 to the interpreter and Figure 6 for the same negative authorization policy of Listing 4.

Listing 4. Negative authorization policy example

```
inst auth− researcherOps {
   subject <User>      /Researchers;
   target  <OFSwitch>  /Nregion/switches;
   action enable(), disable();
}
```

In Listing 4, we define a negative authorization policy restricting users from the researchers group does not perform the actions to enable or disable the switches in a Nregion.

The language also provides the ability to define policy types, enabling the reuse of policies by passing formal parameters in its definition. Several instances of the same type can be created and adapted to the identical environment through real values as arguments.

Listing 5. Type definition policy sintax

```
type ( auth+ | auth− ) policyType ( formalParameters) {
   authorization−policy−parts
}
inst ( auth+ | auth− ) policyName = policyType(
      actualParameters )
```

The authorization policy switchPolicyOps (from Listing 2) can be specified as a type of the subject and target given as parameters as shown in Listing 6.

Listing 6. Type policy definition example

```
type auth+ PolOpsT(subject s, target <OFSwitch> t) {
     action load(), remove(), enable(), disable() ;
}
inst auth+ admPolyOps=PolOpsT(/NetworkAdmins,
      /NregionA/switches);
inst auth+ rsrPolOps=PolOpsT( /Researchers,
      /NregionB/switches);
```

Furthermore, we can use the PonderFlow Authorization Policies to define a flow in the OpenFlow network. A flow

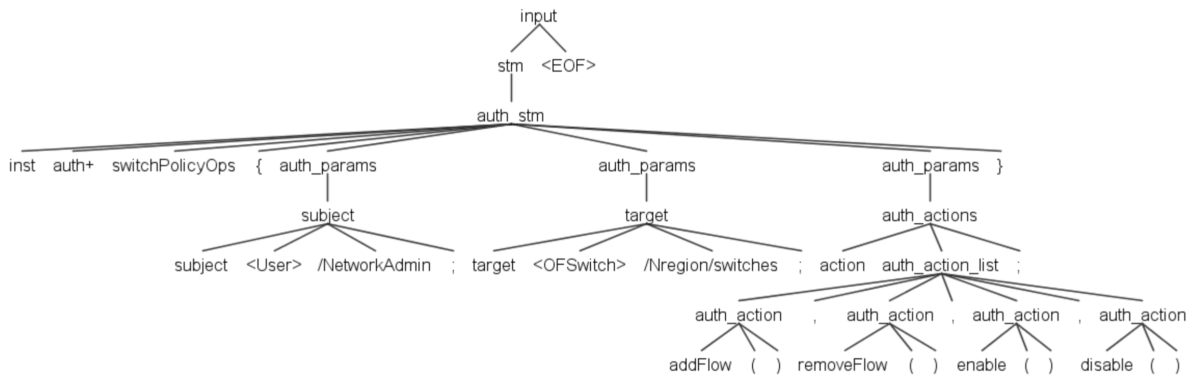


Figure 5. The AST tree for Listing 2 example

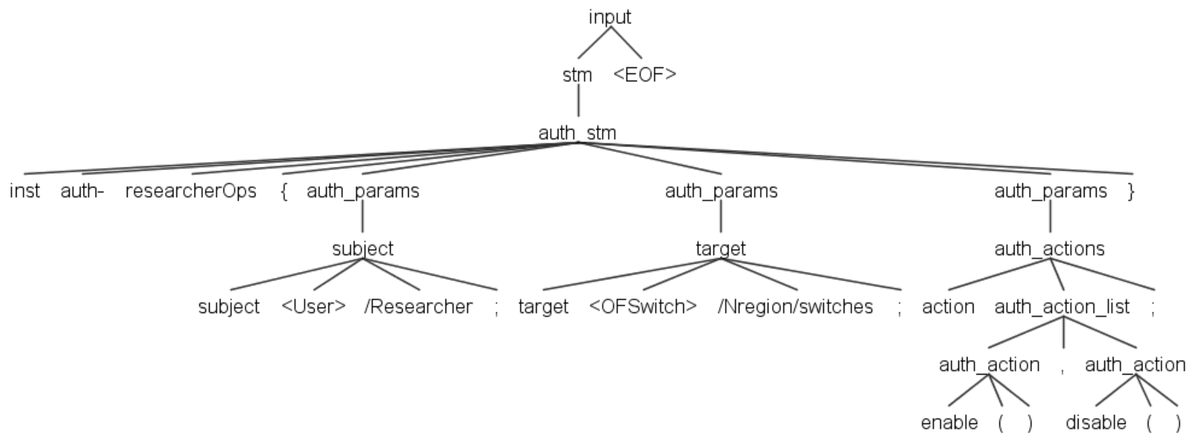


Figure 6. The AST tree for Listing 3 example

is an OpenFlow network path between hosts, independent of the switch quantity.

Thus, network administrator does not need to use a programming language like Java, Python or C++, in order to manipulate directly behavior through of the OpenFlow controller.

Listing 7. Type policy definition example

```
flow-expression = on = <DPID> ,
          | src = <DPID>/<switch_port> ,
          | src = <IP-ADDRESS> ,
          | src = <MAC_ADDRESS> ,
          | dst = <DPID>/<switch_port> ,
          | dst = <IP-ADDRESS> ,
          | dst = <MAC_ADDRESS> ,
          | by = <DPID> ;
```

To define a flow, we need to use the keyword **flow** in the authorization policy statement. With this keyword, we can define the characteristic of the flow. Furthermore, it is possible define a path restriction where the network administrator can define where the flow must pass.

Listing 7 shows the grammar of *flow-expression*, where: *DPID* is the switch identification, *src* and *dst* are respectively the source device and destination device, *switch_port* is the incoming packet switch port, *IP-ADDRESS* is a valid IP address and *MAC-ADDRESS* is a valid MAC address.

TABLE I. OPENFLOW POLICY WILDCARDS

ingress-port	The switch port on which the packet is received
src-mac	The source mac address value
dst-mac	The destination mac address value
vlan-id	The VLAN identification value
vlan-priority	The VLAN priority value
ether-type	The ethernet type value
tos-bits	The ToS bits value
protocol	The IP protocol number used in the protocol field
src-ip	The source IP address value
dst-ip	The destination IP address value
src-port	The source protocol port value
dst-port	The destination protocol port value

The example in Listing 8 authorizes a flow to user /User/Students/John (**subject**), on the switches of domain */Uece/Macc/Larces/Switches*, set flows (**action**) on the network to establish a path starting from the switch with Datapath ID (DPID) 00:00:00:2C:AB:7C:07:2A on the port 2 (**src**) and ending in the switch with DPID 00:00:00:47:5B:DD:3F:1B on port 5 (**dst**), passing by the switches with DPID 00:00:00:C5:FF:21:7F:3B and 00:00:00:33:45:AF:1C:8A (**by**) when the source IP address of the flow is 192.168.0.21, the destination IP address 192.168.0.57 and the protocol destination port is 80.

Listing 8. A PonderFlow authorization policy

```
inst auth+ flow01{
  subject <User>    /Users/Students/John;
  target  <Switch> /Uece/Macc/Larces/Switches;
  flow <Flow> src=00:00:00:2C:AB:7C:07:2A/2  ,
              dst=00:00:00:47:5B:DD:3F:1B/5  ,
              by  =00:00:00:C5:FF:21:7F:3B  ,
                  00:00:00:33:45:AF:1C:8A  ;
  action setFlow();
  when src-ip=192.168.0.21,
       dst-ip=192.168.0.57,
       dst-port=80;
}
```

PonderFlow specifies a set of default actions for flow definition, but the developers are free to add more actions to the language. The default actions are listed in Table II. Listing 9 defines a policy which user Alice can set a flow action to change the source IP address of the packet to 10.23.45.65 when the destination IP address is 10.23.45.123 on the switch with DPID 00:00:00:4F:32:1D:56:9C.

Listing 9. The flow definition to change the source ip address

```
inst auth+ flow02{
  subject <User>    Alice;
  target  <Switch> 00:00:00:4F:32:1D:56:9C;
  action setSrcIP('10.23.45.65');
  when dst-ip=10.23.45.123;
}
```

Furthermore, it is possible to define a policy to be applied in a specific switch and not a path. This is desirable when the network administrator wishes to add or remove a particular flow in a specific switch, in this way, the network administrator changes the network behavior in a single point on the network.

In Listing 10 is shown a policy example authorizing user Bob adds a flow (**action**) for the (**subject**) in the switch 00:00:00:4F:32:1D:56:9C (**target**) when the destination IP address is 172.24.5.17, and the destination port is 5432.

Listing 10. Authorizing add a specific flow in the switch

```
inst auth+ flow3{
  subject <User>    Bob;
  target  <Switch> 00:00:00:4F:32:1D:56:9C;
  action setFlow();
  when dst-ip=172.24.5.17,
       dst-port=5432;
}
```

TABLE II. OPENFLOW ACTION FIELD

setFlow()	Set the flow(s) in a specified path
delFlow()	Delete the flow(s) in a specified path
setSrcIp(ip-address)	Set the source IP address of the packet
setDstIp(ip-address)	Set the destination IP address of the packet
setSrcMac(mac-address)	Set the source MAC address of the packet
setDstMac(mac-address)	Set the destination MAC address of the packet
setSrcPort(port)	Set the source port of the packet
setDstPort(port)	Set the destination port of the packet
setVlanId(integer)	Set the VLAN of the packet
setVlanPriority(integer)	Set the VLAN priority of the packet

We can restrict the actions of the network users with the authorization policies. For example, we cannot permit a certain kind of flows through the OpenFlow network with the authorization policies. In Listing 11 we define an authorization policy not allowing Alice (**subject**) add flows (**action**) in the switches with DPID 00:00:00:4F:32:1D:56:9C, 00:00:00:47:5B:DD:3F:1B and 00:00:00:33:45:AF:1C:8A.

Listing 11. Negative authorization policy for deny add flow in the switch

```
inst auth- flow4{
  subject <User>    Alice;
  target  <Switch> /Uece/Macc/Larces/Switches;
  flow by=00:00:00:4F:32:1D:56:9C,
          00:00:00:47:5B:DD:3F:1B,
          and 00:00:00:33:45:AF:1C:8A
  action setFlow();
}
```

Another kind of usage is not allows the user to change a packet values in the network. Listing 12 shows an example of this policy.

Listing 12. Negative authorization policy for deny add flow in the switch

```
inst auth- flow5{
  subject <User>    Alice;
  target  <Switch> /Uece/Macc/Larces/Switches;
  action setSrcIP(),
         setDstIp(),
         setSrcMac(),
         setDstMac(),
         setSrcPort(),
         setDstPort();
}
```

The previous example the user Alice (**subject**) cannot change the source ip address, destination ip address, source mac address, destination mac address, source port and destination port (**action**) in any switch of */Uece/Macc/Larces/Switches* domain.

Moreover, it is possible define action which specific user can perform in the network. Listing 13 shows a policy which allows the user of domain */Uece/Macc/Larces/Admin* (**subject**) remove flows (**action**) of in any switch of */Uece/-Macc/Larces/Switches* domain (**target**).

Listing 13. Authorization policy for allow remove flow

```
inst auth- flow5{
  subject <User>    /Uece/Macc/Larces/Admin;
  target  <Switch> /Uece/Macc/Larces/Switches;
  action delFlow();
}
```

B. Obligation Policies

Obligation policies allow to specify actions to be performed by the network administrator or by the OpenFlow controller when certain events occur in an OpenFlow network and provide the ability to respond any change in circumstances.

These policies are event-triggered and define the activities subjects (network administrator or OpenFlow controller) must perform on objects within the target domain. Events can be simple, e.g., an internal timer, or more complex, starting by reading some kind of sensor, e.g., a network card stopped.

This building block is very similar to the original language Ponder, but in the context of PonderFlow, including flow definition. This block sets an obligation for the network administrator or the OpenFlow controller performs some action, or simply is notified, when a particular event occurs.

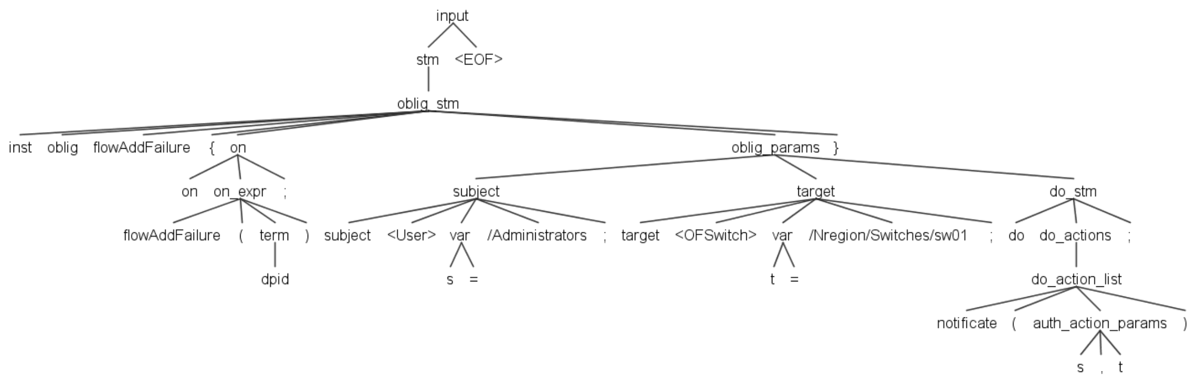

input
stm <EOF>
oblig_stm
inst oblig flowAddFailure { on
oblig_params }
on on_expr ;
subject
target
do_stm
flowAddFailure (term)
subject <User> var /Administrators ;
target <OFSwitch> var /Nregion/Switches/sw01 ;
do do_actions ;
dpid
s =
t =
do_action_list
notificate (auth_action_params)
s , t

Figure 7. The AST tree for Listing 16 example

Listing 14. Obligation policy sintax

```
inst oblig policyName {
  on event−specification ;
  subject [<type_def >] domain−Scope−Expression ;
  [ target [<type_def >] domain−Scope−Expression ; ]
  do obligation−action−list ;
  [ catch exception−specification ; ]
  [ when constraint−Expression ; ]
}
```

The syntax of obligation policies is shown in Listing 14. The required event specification follows the **on** keyword. The target element is optional in obligation policies. The optional catch-clause specifies an exception to be performed if the actions fail to execute, for some reason.

A snippet of ANTLR grammar defined for PonderFlow obligation policies is described in Listing 15.

Listing 15. Obligation policy grammar

```
oblig_policy:
  INST OBLIG policy_name
    OPEN_BODY
      oblig_policy_options*
    CLOSE_BODY;
```

In Listing 16, the obligation policy is triggered when a failure on adding a flow occurs. Network administrator will be notified when this event occurs, and he will receive the switch ID where it happened. Figure 7 shows the AST tree of Listing 16.

Listing 16. Obligation policy sintax

```
inst oblig flowAddFailure {
  on flowAddFailure(dpid) ;
  subject <User> s=/Administrators ;
  target <OFSwitch> t = /Nregion/Switches/sw01 ;
  do notificate(s, t) ;
}
```

To perform an obligation policy, it is required the user has an authorization over the target. This can be specified with an authorization policy. If there is no authorization policy specifying who can perform a particular action, the obligation policy will produce an exception error (depends on the implementation), and the policy will not be applied in the system.

V. PonderFlow Parser

A PonderFlow parser was developed in Java using ANTLR framework [26]. The ANTLR framework is a flexible and powerful tool for parsing formal languages like PonderFlow. The parser translates the PonderFlow statements describe in Section IV.

ANTLR provides support for two tree-walking mechanisms; the parse-tree listeners and the parse-tree visitors. By default, ANTLR generates a parse-tree listener interface to respond to events executed by the built-in tree walker. The PonderFlow parser uses the parse-tree listeners to parse its grammar.

The parser consists of some Java classes, generated automatically by ANTLR. The PonderFlow parser is basically a Java class implementing a parse-tree listener interface, and this interface requires some Java methods to be implemented.

In these methods, we add Java code describing what must be done for each PonderFlow statement read. It is possible to use other tools such as a DBMS or other framework to assist in analyzing the statements OpenFlow process.

Some performance test was executed over PonderFlow parser. The instances of the performance testing vary between 1 and 10000 statements. Figure 8 shows the elapsed time to parse instances of 1, 5, 10, 20, 50, 100, 200, 500, 1000, 2500, 5000 and 10000 PonderFlow statements, respectively.

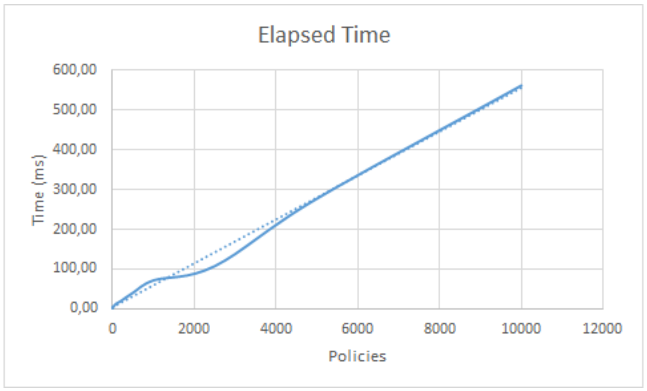


Figure 8. Time required to parse several instances.

As shown in Figure 8 can be seen the analyzer is easily scalable and can analyze tens of thousands of statements in a second. Use other tools, such as a DBMS, the time of analysis may increase, hampering the analysis process.

However, since some compilers of programming languages may take several seconds to compile the source code, even if it is added to other tools and frameworks on the PonderFlow parser, the compile time will still be at a tolerable level of acceptability.

VI. Conclusion and Future Works

This paper described the PonderFlow language, a new policy specification language for OpenFlow networks. With this language, the network administrator does not need to be an expert in a programming language, like Java, Python or C++, to specify the policy of an OpenFlow network. The language statements are simple and concise to define policies.

The PonderFlow grammar was presented as well as some examples of usage and their AST tree representation. The grammar was tested using the ANTLR framework, which generates the parser and the lexical analyzer for the Java programming language.

Some tests were performed and it was observed that this solution had scalability, and is easily integrated into an OpenFlow controller. However, the PonderFlow works merely with OpenFlow Switch Specification version 1.0, because most of the commercial switches only support this version.

It shall extend the Ponder language to use the OpenFlow Switch Specification version 1.3. Another point that should be studied is the treatment of policy's conflicts, where a network administrator can, by accident or malpractice, declare two or more conflicting policies. It is necessary to perform an assessment on all policies before applying them on OpenFlow controller.

References

[1] B. Batista and M. Fernandez, "Ponderflow: A policy specification language for openflow networks," in ICN 2014, The Thirteenth International Conference on Networks, Nice, France, Feb. 2014, pp. 204–209.

[2] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, "OpenFlow: enabling innovation in campus networks," ACM SIGCOMM Computer Communication Review, vol. 38, no. 2, 2008, pp. 69–74.

[3] N. Damianou, N. Dulay, E. Lupu, and M. Sloman, "The ponder policy specification language," in Proceedings of the International Workshop on Policies for Distributed Systems and Networks (POLICY '01). London, UK, UK: Springer-Verlag, 2001, pp. 18–38. [Online]. Available: http://dl.acm.org/citation.cfm?id=646962.712108

[4] N. Foster, R. Harrison, M. J. Freedman, C. Monsanto, J. Rexford, A. Story, and D. Walker, "Frenetic: a network programming language," SIGPLAN Not., vol. 46, no. 9, Sep. 2011, pp. 279–291. [Online]. Available: http://doi.acm.org/10.1145/2034574.2034812

[5] NOXRepo.org, "NOX Openflow Controller," Last accessed, Aug. 2014. [Online]. Available: http://www.noxrepo.org/nox/about-nox/

[6] G. VanRossum and F. L. Drake, The Python Language Reference. Python Software Foundation, 2010.

[7] D. M. F. Mattos, N. C. Fern, V. T. D. Costa, L. P. Cardoso, M. Elias, M. Campista, L. H. M. K. Costa, and O. C. M. B. Duarte, "Omni: Openflow management infrastructure," Paris, France, 2011.

[8] A. Voellmy, H. Kim, and N. Feamster, "Procera: a language for high-level reactive network control," in Proceedings of the first workshop on Hot topics in software defined networks, ser. HotSDN '12. New York, NY, USA: ACM, 2012, pp. 43–48. [Online]. Available: http://doi.acm.org/10.1145/2342441.2342451

[9] D. Erickson, "Floodlight Java based OpenFlow Controller," Last accessed, Aug. 2014. [Online]. Available: http://floodlight.openflowhub.org/

[10] ——, "Beacon," Last accessed, Jun. 2014. [Online]. Available: https://openflow.stanford.edu/display/Beacon/Home

[11] NOXRepo.org, "POX Openflow Controller," Last accessed, Aug. 2014. [Online]. Available: http://www.noxrepo.org/pox/about-pox/

[12] NEC Corporation, "Trema Openflow Controller," Last accessed, Aug. 2014. [Online]. Available: http://trema.github.com/trema/

[13] B. Heller, "Openflow switch specification, version 1.0.0," Dec. 2009. [Online]. Available: www.openflowswitch.org/documents/openflow-spec-v1.0.0.pdf

[14] OpenFlow Hub, "Indigo OpenFlow Switching Software Package," Last accessed, Jun. 2013. [Online]. Available: http://www.openflowswitch.org/wk/index.php/IndigoReleaseNotes

[15] O. Ferkouss, I. Snaiki, O. Mounaouar, H. Dahmouni, R. Ben Ali, Y. Lemieux, and O. Cherkaoui, "A 100gig network processor platform for openflow," in Network and Service Management (CNSM), 2011 7th International Conference on. IEEE, 2011, pp. 1–4.

[16] R. Ozdag, "Intel ethernet switch fm6000: SDN with openflow," Intel Corporation, Tech. Rep., 2012.

[17] NoviFlow Inc, "NoviFlow Switch 1.1," Last accessed, Sep. 2013. [Online]. Available: http://www.noviflow.com/

[18] R. Bertó-Monleón, E. Casini, R. van Engelshoven, R. Goode, K.-D. Tuchs, and T. Halmai, "Specification of a policy based network management architecture," Military Communication Conference, 2011, pp. 1393–1398.

[19] D. C. Verma, "Simplify network administration using policy-based management," IEEE Network, vol. 16, no. 2, March/April 2002, pp. 20–26.

[20] M. Sloman, "Policy driven management for distributed systems," Journal of Network and Systems Management, vol. Vol.2, no. No 4, 1994.

[21] N. C. Damianou, A. K. Bandara, M. S. Sloman, and E. C. Lupu, "A survey of policy specification approaches," April 2002.

[22] E. C. Lupu and M. Sloman, "Conflicts in policy-based distributed systems management," IEEE Trans. Softw. Eng., vol. 25, no. 6, Nov. 1999, pp. 852–869. [Online]. Available: http://dx.doi.org/10.1109/32.824414

[23] B. L. A. Batista, G. A. L. de Campos, and M. P. Fernandez, "A proposal of policy based OpenFlow network management," in 20th International Conference on Telecommunications (ICT 2013), Casablanca, Morocco, May 2013.

[24] K. Twidle, E. Lupu, N. Dulay, and M. Sloman, "Ponder2-a policy environment for autonomous pervasive systems," in Policies for Distributed Systems and Networks, 2008. POLICY 2008. IEEE Workshop on. IEEE, 2008, pp. 245–246.

[25] T. Phan, J. Han, J.-G. Schneider, T. Ebringer, and T. Rogers, "A survey of policy-based management approaches for service oriented system," 19th Australian Conference on Software Engineering, 2008.

[26] T. Parr, "ANTLR: ANother Tool for Language Recognition," Last accessed, Aug. 2013. [Online]. Available: http://www.antlr.org/

Energy Based Analytical Modelling of ANCH Clustering Algorithm for Wireless Sensor Networks

Morteza M. Zanjireh, Hadi Larijani, and Wasiu O. Popoola
School of Engineering and Built Environment
Glasgow Caledonian University
Glasgow, UK
{Morteza.Zanjireh, H.Larijani, Wasiu.Popoola}@gcu.ac.uk

***Abstract*—Wireless Sensor Networks (WSNs) have had remarkable advances in the past couple of decades due to their fast growth and flexibility. In order to supervise an area, hundreds or thousands of sensors can be established and collaborate with each other in the environment. The sensors' sensed and collected data can be delivered to the base station. Energy optimisation is crucial in WSN's efficiency. Organising sensor nodes into small clusters helps save their initial energy and thus increases their lifetime. Also, the number and distribution of Cluster Heads (CHs) are fundamental for energy saving and flexibility of clustering methods. Avoid Near Cluster Heads (ANCH) is one of the most recent energy-efficient clustering algorithms proposed for WSNs in order to extend their lifetime by uniform distributing of CHs through the network area. In this manuscript, we suggest an analytical approach to model the energy consumption of the ANCH algorithm. The results of our comprehensive research show a 95.4% to 98.6% accuracy in energy consumption estimation using the proposed analytical model under different practical situations. The suggested analytical model gives a number of indications concerning the impact of different factors on the energy depletion pattern of the ANCH clustering algorithm.**

***Keywords*—Wireless Sensor Networks, ANCH, Energy Efficiency, Clustering, Analytical Model.**

I. Introduction

A Wireless Sensor Network (WSN) is a network of tiny and on-board battery operated sensors with limited power of processing and radio data transmission. They can collect and send their sensed data to a base station to monitor a remote area and perhaps to send the collected data to a remote centre. WSNs can be employed in different applications, because of their low-cost and adaptable nature, including healthcare, emergency response, weather forecasting, commercial, smart traffics, surveillance and volcanic earthquakes [1]–[6]. Moreover, WSNs can be used in an ad-hoc manner and in harsh environments in which a human presence is hard or impossible [7], [8].

Energy efficiency is essential for WSN lifetimes because there is usually no opportunity for battery replacement or recharging. Therefore, developing energy-efficient algorithms is of high importance in WSNs. A large amount of research has been conducted over the past few years to optimise the energy consumption in this area [9]–[12].

Clustering is a widely accepted approach for organising a high number of sensors spread over a large area in an ad-hoc manner [13]. This is useful when we consider that in most cases, neighbouring sensors sense similar data. If each sensor directly sends its data to the base station using long-distance transmission, its energy drains quickly. Moreover, this might also lead to some other issues, such as traffic congestion and data collision. In clustering, all sensors are grouped into a number of clusters and in each cluster, one sensor is elected as the Cluster Head (CH). Transmitting data in each cluster from each sensor to its CH is accomplished using short-distance radio transmission, whereas transmitting data from CHs to the base station is carried out using long-distance radio transmission. In addition, CHs can combine and aggregate similar transmitted data to reduce the length of the messages passed. In this way, CHs consume more energy than other sensors and, therefore, periodic CH election across the network is a suitable approach to balance energy consumption among sensor nodes [14].

The appropriate number and size of the clusters is essential for increasing the network lifetime. For a low number of clusters, a large amount of the energy is consumed to send data from Cluster Members (CMs) to CHs. On the other hand, if the number of clusters is high, a large number of the CHs will be elected and consequently a large number of nodes will operate using long-distance transmission to communicate with the base station. Therefore, a trade-off should be made between these two factors to optimise energy consumption across the network [15].

Over the past few years, a number of clustering algorithms have been proposed. Hence, it is critical that when proposing a new algorithm, we specify its scope and evaluate it with accurate modelling of the underlying organisation and communication mechanisms. Clearly, after using such models, a comprehensive understanding of the factors that affect the potential performance of a network emerges and this makes it easier to evaluate different algorithms and select the best one for practical implementations. Employing physical experiments is impractical for a large number of configurations and running a network simulator for a large number of configurations takes an unacceptable amount of time. Analytical modelling, in contrast, offers a cost-effective and versatile tool that can help to assess the merits of an algorithm [16], [17].

Avoid Near Cluster Heads (ANCH) is a new energy efficient clustering algorithm proposed recently for WSNs to prolong network lifetimes by uniformly distributing the CHs [12]. In this manuscript, an analytical model for predicting the energy consumption of ANCH is proposed. The model details the factors affecting and analyses energy consumption under various operational conditions. The accuracy of the proposed model is evaluated using simulation.

The remainder of this manuscript is organised as follows. In Section II, related work is discussed. The ANCH clustering algorithm is briefly presented in Section III. The proposed analytical model of ANCH and its validation are presented in Section IV and Section V, respectively. Finally, Section VI contains our concluding remarks and future work.

II. Related Work

Over the past few years, a number of clustering algorithms for WSNs have been proposed such as Low Energy Adaptive Clustering Hierarchy (LEACH) [9], Hybrid Energy-Efficient Distributed (HEED) [18], Low Energy Adaptive Clustering Hierarchy with Sliding Window and Dynamic Number of Nodes (LEACH-SWDN) [19], ANCH [12], and Activity-aware ANCH (A-ANCH) [20]. One of the most popular clustering algorithms for WSNs is LEACH. LEACH is popular not only for its simplicity, but also for the idea of rotating CHs to efficiently balance energy consumption among nodes [9]. The lifetime of the LEACH is divided into a number of rounds in which every round includes a set-up and a steady phase. In set-up phase, clusters are organised, while in steady phase, sensed data are transferred from sensors to CHs. Election of CHs is carried out in a distributed manner and independent from each other. CHs collect sensed data from sensors and aggregate and forward them to the base station in each round. To decide whether to become a CH, each sensor chooses a random number x, which is a number between 0 and 1, in each round. A sensor decides to be a CH in a round if its generated random number is less than the threshold below [9]:

$$T\left(n\right)=\begin{cases}\frac{p}{1-p\left(\text{r mod }\frac{1}{p}\right)} & \text{if } n\in G\\ 0 & \text{otherwise}\end{cases} \tag{1}$$

where n is the number of each sensor in the network, r is the current interval's round number, p is a predefined percentage of CHs in each round, $\frac{1}{p}$ is called the interval, and G is the set of sensors which have not yet been CHs in the current interval. An example of CHs and CMs formed after a set-up phase in a round according to the LEACH algorithm is shown in Figure 1.

In LEACH, p percent of sensors are elected as CHs on average in each round and also each sensor is elected as a CH only once in an interval. The probability of each node becoming a CH is p in the first round. This probability is $p/(1-p)$ and $p/(1-2p)$ for the second and third rounds and finally this probability reaches 1 for the last round. Thus all of the remaining sensors will be CHs in the last round. Each sensor that has decided to be a CH in a round, advertises itself

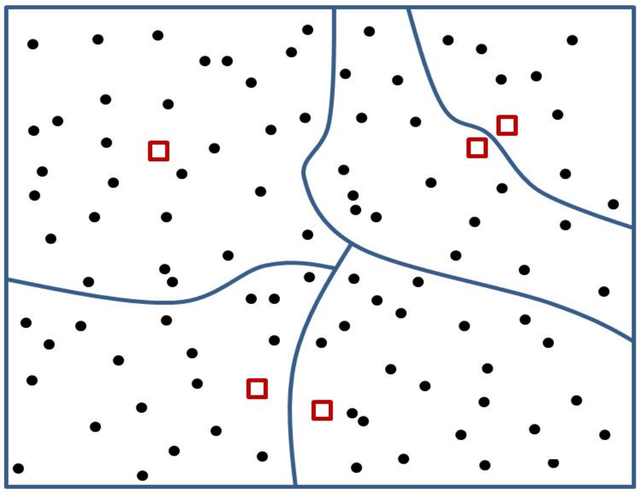


Fig. 1: An example of CHs and CMs arrangement in the LEACH algorithm. CHs are shown by red diamonds and CMs are shown by black circles.

to all of the other sensors throughout the network. Using the received signal's strength, all non-CH sensors can find the closest CH to themselves and join that cluster as a CM.

A centralised version of LEACH, called LEACH-C, is proposed by the authors of LEACH, Heinzelman *et al.* [21], in which each sensor knows its position in the network and consequently the number and position of CHs are selected in an optimum manner per round. It has been demonstrated that LEACH-C shows up to 40% greater performance than LEACH. Another centralised version of LEACH is Base station Controlled Dynamic Clustering Protocol (BCDCP), which was proposed by Muruganathan *et al.* [22]. BCDCP uses multi-level clustering so that each CH serves an approximately equal number of sensors. In the BCDCP algorithm all CMs send their sensed data to their CHs and CHs conduct a CH-to-CH multi-hop routing to deliver their data to a higher level CH, which is selected randomly. Finally, only one CH communicates directly with the base station. It has been shown that BCDCP in this way outperforms LEACH and LEACH-C.

Two distributed variations of LEACH have been proposed by Nam *et al.* [23] and Handy *et al.* [24]. In the first algorithm, a number of messages are exchanged between sensors to find out their position across the network. Thus, more equal clusters are shaped by choosing appropriate CH positions. They compared their algorithm with LEACH and LEACH-C and their experiments showed that their algorithm outperforms LEACH but not LEACH-C, which is a centralised algorithm.

On the other hand, Handy *et al.* [24] believe that a stochastic CH election might not lead to an efficient energy consumption model in a network. Therefore, they involved the sensor's residual energy when computing the threshold $T(n)$ as an effective parameter in deciding whether or not to stand as a CH. Their experiments showed that their algorithm can prolong network lifetime by up to 30%.

Heterogeneity of sensors energy in the network has been treated by Smaragdakis *et al.* [25]. They proposed the Stable Election Protocol (SEP) algorithm, in which a number of sensors are equipped with more energy than others to prolong the stable time of the network, which is when all the network's sensors are alive. SEP is a distributed algorithm and sensors are not aware of other sensors' residual energy. Their experiments showed that SEP can increase the stable time of the network and this increment depends on the percentage and the initial energy levels of the network's powerful sensors.

Two centralised fuzzy algorithms for WSNs have been proposed by Gupta *et al.* [26] and Ran *et al.* [27]. They assumed that each sensor knows its position in the network and sends its local information to the base station. Appropriate sensors are then elected by the base station to act as CHs. In both algorithms, three parameters are used as the algorithm input. In [26] these parameters are *sensor residual energy, sensor density*, and *sensor centrality* in the clusters whereas in [27] the *sensor centrality* is replaced by *distance of sensors from the base station*. The performance evaluations of both studies showed the superiority of their algorithms over the LEACH clustering algorithm.

Wireless Sensor Network Clustering with Artificial Bee Colony (WSNCABC) has been proposed by Karaboga *et al.* [28]. In WSNCABC each sensor calculates its distance from other sensors using message transferring advertisement and sends this information to the base station. The base station then selects CHs using an Artificial Bee Colony algorithm and broadcasts them back to the network. Finally, each sensor joins the closest elected CH. Their performance study showed that the proposed algorithm considerably outperforms the LEACH clustering algorithm, by up to 70% in some scenarios, when the base station is very close to the edge of the network. However, as there is a large amount of communication between sensors and the base station in the set-up phase, WSNCABC appears to be inefficient when the base station is located far away from the network area.

HEED [18] is a distributed clustering algorithm for WSNs which takes into account sensors residual energy and communication cost during CH election. In HEED, the transmission power of every node is set to a constant value and each sensor considers other nodes as its neighbouring nodes if they are within its transmission range. Moreover, two neighbouring sensors, which are within the transmission range of each other, are not elected as CHs simultaneously, trying to uniformly distribute CHs across the network. In order to find its CH, each sensor launches an iterative procedure in each round. A node nominates itself as a CH if it has either no neighbouring node or has not received a CH advertisement.

ANCH also, similar to HEED, takes the advantage of uniform distribution of CHs in order to achieve optimised, or close to, network energy consumption. Nevertheless, it has a few key advantages over HEED. Firstly, the set-up phase overhead of ANCH is much less than that of HEED because HEED executes a procedure to find neighbouring sensors. Also, in this phase, each sensor in HEED executes a complicated iteration including some message passing to select its CH. Secondly, by the end of each iteration in HEED, a node elects itself as a CH if no other CH advertisement has been received. Thus, in many rounds, the number of formed clusters is much more than that of the ANCH algorithm where all sensors receive CH advertisement if there exists at least one CH in the network. Finally, ANCH and LEACH are two scalable algorithms both with processing time and message exchange complexity of O(1) and O(N), respectively [29]. Whereas, HEED has O(N) complexity for both processing time and message exchange complexity [30], [31].

In order to design an energy efficient algorithm for WSNs, it is important to make a trade-off between different parameters involved in a specific application to ensure that the optimum configuration has been applied to maximise the network lifetime. In particular, it is quite critical to balance the energy costs of individual nodes in order to obtain the best overall network energy cost. Simulation study of the effects of different parameters on the performance of a network under various network circumstances is difficult because of the time consuming feature of these kinds of tools. Analytical modelling, in contrast, is beneficial as it offers a cost-effective tool to estimate the network energy consumption accurately within an acceptable amount of time. Therefore, in addition to the research on proposing efficient algorithms for WSNs, a number of studies have also been conducted to develop analytical models [15]–[17], [21], [32].

The first analytical model for the LEACH algorithm has been proposed by Heinzelman *et al.* [21]. In this study, it has been shown that the energy consumption in a network is proportional to the square of transmission distance in clusters. This can be obtained for each sensor using the following expression:

$$E\left[d_{toCH}^2\right] = \rho \int_{\theta=0}^{2\pi} \int_{r=0}^{\frac{M}{\sqrt{\pi k}}} r^3 dr d\theta = \frac{M^2}{2k\pi} \tag{2}$$

where $E\left[d_{toCH}^2\right]$ is the expected square distance of sensors from their CH, $\rho = \frac{k}{M^2}$ and is called sensors' density, k is the number of clusters, and M is one side of the network area.

However, some non-realistic assumptions have been made when developing the model; the area of all clusters are disc-shaped with radius r, all clusters are assumed to be formed equally, and also the area of the network is covered by these k non-overlapping clusters.

In [33], Bandyopadhyay and Coyle have proposed a mathematical model for hierarchical clustering algorithms for WSNs. They assumed that the sensors are very simple and all sensors transmit at a fixed power level. Their model analytically suggests the number of CHs at each level of clustering. They conducted a set of experiments to show the optimum number of CHs in different levels of hierarchy in dense networks, with up to 25,000 nodes. Nevertheless, their proposed model is not general enough due to a number of unrealistic assumptions on the fixed power level transmitting ability of nodes.

In [15], Islam *et al.* have argued that a complete analytical model to find the best configuration on LEACH is beneficial to prolong network lifetimes. They therefore proposed a complete mathematical model for LEACH to find the optimum number of CHs. Their experiments showed the reasonable accuracy of their model against original LEACH in order to compute the optimal number of CHs. Although, their model takes into account all steps of the LEACH algorithm, it does not show a considerable advantage over the model presented in [21].

The issues raised in [21] have been addressed in another model proposed by Choi *et al.* [34]. This model can estimate the energy consumption and determine the optimal number of clusters to minimise the energy spent in the network for the LEACH algorithm. In this research, it has been shown that the energy consumption in a network is proportional to the summation of square of transmission distance in clusters. This can be obtained for each cluster using the following equation:

$$E\left[\sum_{node\in cluster(j))} d^2_{toCH}\right] = 2\pi\lambda_{CM}\int_0^\infty r^3 exp(-\lambda_{CH}\pi r^2)dr \qquad (3)$$

where $E\left[\sum_{node\in cluster(j))} d^2_{toCH}\right]$ is the expected sum of the square distance of sensors from their CH, λ_{CM} is the density of the CMs in the network area and is given by $\frac{N-k}{M^2}$, λ_{CH} is the density of the CHs in the network area and is given by $\frac{k}{M^2}$, N is the number of nodes, k is the number of clusters, and M is one side of network area.

Using a simulation software, their experiments showed that their model has over 80% accuracy with the LEACH algorithm and is considerably superior to [21].

Details of the ANCH algorithm are described in the next section.

III. The ANCH Clustering Algorithm

The proper position of CHs is essential in the energy efficiency of clustering algorithms. This has been neglected in the LEACH algorithm and consequently there might be some CHs which are located too close or too far from each other. In either case, some waste of energy might occurred for data transferring from sensors to the base station.

To overcome this, the ANCH algorithm tries to uniformly distribute CHs across the network as much as possible. To do so, a parameter d is defined as the *closeness* depending on the region size and also network density. If two CHs are found too close to each other in a particular round, closer than d, one of them should stand as the CH. Thus once the first CH is selected following normal LEACH procedure, the next potential CH checks its distance from the first CH before advertising itself to other sensors as a CH. If the distance is less than d, it cancels its decision to be a new CH in the current round and remains a CH candidate for the future rounds. The same procedure is applied for all of other potential CHs in the current round.

This action is applied and can be considered as the first attempt to shape clusters with about the equal size. Figure 2 demonstrates an example of CH selection according to the first step of the ANCH algorithm.

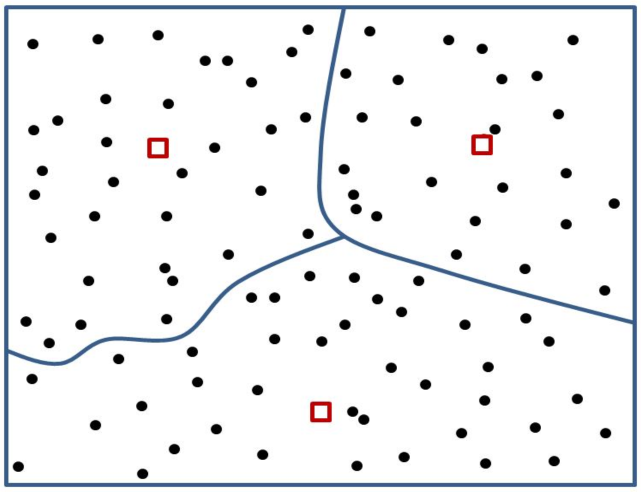


Fig. 2: An example of CHs and CMs in first step of ANCH algorithm. CHs are shown by red diamonds and CMs are shown by black circles.

Also, at the end of each interval, the number of the CHs is most likely more than the optimum amount. Thus, the number of clusters probably increase and more CHs have to send their packets to the base station by using long-distance communication. Figure 3 demonstrates an example of CHs and CMs at the end of each interval.

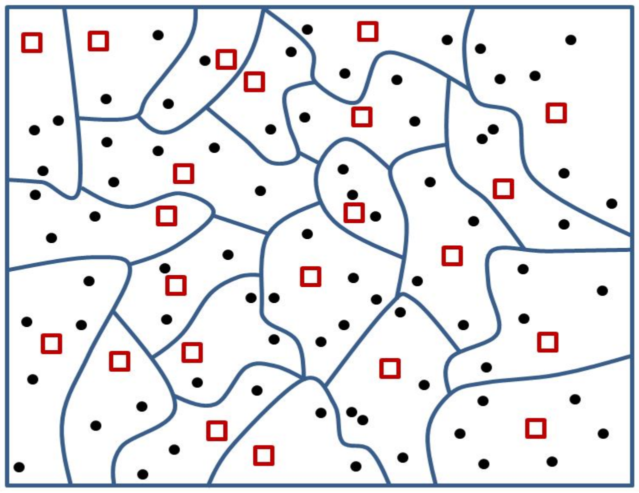


Fig. 3: An example of CHs and CMs at the end of each interval in the ANCH algorithm. CHs are shown by red diamonds and CMs are shown by black circles.

Since all sensors have to be a CH in an interval, all of the remaining sensors will be elected as CHs in the last round regardless of their closeness to each other.

A further improvement in ANCH is also obtained by considering the optimum number of CHs in the network. Imagine p is the optimum percentage of CHs among all sensors. So far in the ANCH algorithm, apart from the intervals' last rounds, the number of selected CHs in each round is most likely less than p percent. This is because a number of potential CHs might cancel their decision of being a CH due to their close position to other CHs. Therefore, the number of clusters would be less than the optimum number suggested in the LEACH algorithm. This leads to the bigger cluster size and more energy consumption over the intra-cluster transmission.

On the other hand, in the last rounds of intervals, the percentage of CHs would be much more than the optimum number p. Consequently, the number of clusters might increase and more CHs have to send their data to the base station directly using long-distance transmission. This issue is addressed in the ANCH algorithm by increasing the threshold $T(n)$ and consequently increasing the number of potential CHs in each round. As a result, in every round more than p percent of sensors will be nominated as CHs, on average, to become closer to the optimum value, p, after dropping a number of them because of the *closeness* issue. After setting the new threshold, close to p percent of sensors are eventually selected as the CHs in every round which are more uniformly distributed compared with LEACH. The new threshold, $T'(n)$, in ANCH is defined as follows:

$$T'(n) = T(n) + (1 - T(n)) \times a. \tag{4}$$

$T(n)$ is the threshold value of the LEACH algorithm [9] and a, the add-on coefficient, is a constant, whose value depends on the network configuration and also on the *closeness* value, d. This value plays an essential role in the ANCH algorithm's efficiency.

The ANCH algorithm significantly improves network energy consumption and, consequently, prolongs the network lifetime compared with the LEACH algorithm. An example of the positions of CHs and CMs in ANCH is shown in Figure 4. Comparing this arrangement with the one presented in Figure 1 reveals more uniform distribution of CHs in the ANCH algorithm.

The analytical model of the ANCH clustering algorithm is proposed in the next section.

IV. ANCH Analytical Modelling

In this section, our proposed analytical model for the energy consumption in the ANCH clustering algorithm is presented. Using the model, a comprehensive understanding of the factors affecting the performance of a network emerges. Since a clustering approach is employed in the ANCH algorithm, the total network energy consumption can be derived when the energy consumed by one cluster is calculated.

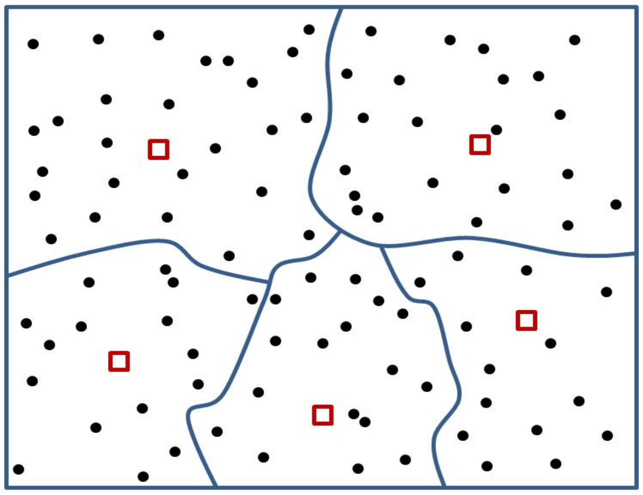


Fig. 4: An example of CHs and CMs arrangement in the ANCH algorithm. CHs are shown by red diamonds and CMs are shown by black circles.

Let us assume that N sensor nodes are randomly distributed in a $M \times M$ area and the number of clusters, on average, is k during the lifetime of the network. As a result, there are $\frac{N}{k}$ sensors, on average, per cluster with $(\frac{N}{k}) - 1$ sensors as CMs and also one node as the CH.

The energy required for a CM to send its data to a CH can be calculated using the following expression [9]:

$$E_{CM} = lE_{elec} + l\epsilon_{amp}d_{toCH}^2 \tag{5}$$

Also, for all nodes in a cluster, this energy can be calculated as follows:

$$\begin{aligned} E_{Cluster} &= lE_{elec}(k-1) \\ &+ l\epsilon_{amp}E\left[\sum_{nodes \in Cluster} d_{toCH}^2\right] \end{aligned} \tag{6}$$

where l is the length of messages, E_{elec} is the transmission electronics, ϵ_{amp} is the transmission amplifier, d_{toCH} is the distance between a CM and its CH, and $E\left[\sum d_{toCH}^2\right]$ is the expected sum of square distance of CMs from their CH. Except for $E\left[\sum d_{toCH}^2\right]$, all other parameters in (6) are known with constant values. Therefore, by calculating $E\left[\sum d_{toCH}^2\right]$ we are able to calculate all the energy spent in the network.

$E\left[\sum d_{toCH}^2\right]$ can be calculated using the following expression for LEACH [34]:

$$\begin{aligned} E\left[\sum_{node \in cluster(j)} d_{toCH}^2\right] &= 2\pi\lambda_{CM} \times \\ \int_0^\infty r^3.P\left\{(r,j) \in cluster(j)\right\} dr \end{aligned} \tag{7}$$

where λ_{CM} represents the density of the CMs in the network and is given by $\frac{N-k}{M^2}$. $P\left\{(r,j) \in cluster(j)\right\}$ is the probability of a sensor node becoming a member of cluster j. The distance between the node and the head of cluster j is also represented by r. According to [35], $P\left\{(r, j) \in cluster(j)\right\}$ can be derived from the palm distribution as follows:

$$P\left\{(r, j) \in cluster(j)\right\} = exp\left\{-\lambda_{CH}\pi r^2\right\} \tag{8}$$

where λ_{CH} represents the density of the CHs in the network and is given by $\frac{k}{M^2}$. In ANCH, the distance between any two CHs is not less than d. Each cluster area is divided into two different parts, which are treated separately in our model. The first part is the circular area with the radius of $d/2$ from the CH. All sensors in this area securely belong to that cluster. The second area covers those sensors whose distance from the current CH is more than $d/2$. For the first part, (7) with the probability $P\left\{(r, j) \in cluster(j)\right\} = 1$ can be used. Thus, the expected sum of the square distance of CMs, located in the first part of the cluster area, from their CH can be obtained using the following expression:

$$E\left[\sum_{node \in cluster(j)} d^2_{toCH}\right] = 2\pi\lambda_{CM}\int_0^{d/2} r^3 dr \tag{9}$$

On the other hand, all sensors whose distance from other CHs is less than $d/2$ are secure members of other CHs and are not members of the current CH. Thus, $P\left\{(r, j) \in cluster(j)\right\} = 0$ for those nodes. Consequently, the value of (7) for those nodes is 0. To calculate the second part of the cluster area, we must subtract the cluster areas whose nodes' distance from a CH is less than $d/2$.

The second part of each cluster area can be calculated by

$$\begin{gathered} E\left[\sum_{node \in cluster(j)} d^2_{toCH}\right] = \\ 2\pi\lambda_{CM}\int_{R_1}^{\infty} r^3 . P\left\{(r, j) \in cluster(j)\right\} dr \end{gathered} \tag{10}$$

In the above expression, R_1 can be calculated as follows:

$$\pi R_1^2 = k\pi(\frac{d}{2})^2 \Rightarrow R_1 = (\frac{d}{2})\sqrt{k} \tag{11}$$

Using (9) and (10), the first and second parts of each cluster area can be merged. Thus, the expected sum of the square of each CM from its CH can be obtained from the following expression:

$$\begin{gathered} E\left[\sum_{node \in cluster(j)} d^2_{toCH}\right] = 2\pi\lambda_{CM} . \\ \left[\int_0^{d/2} r^3 dr + \int_{\left(\frac{d}{2}\right)*\sqrt{k}}^{\infty} r^3 . exp\left\{-\lambda_{CH}\pi r^2\right\} dr\right] \end{gathered} \tag{12}$$

Also we know that

$$\int_0^{\frac{d}{2}} r^3 dr = \left(\frac{d^4}{64}\right) \tag{13}$$

and we can imagine

$$u = \lambda_{CH} . \pi \tag{14}$$

The expression (10) can be simplified to the following expression:

$$\begin{gathered} \int_{\left(\frac{d}{2}\right)*\sqrt{k}}^{\infty} r^3 . exp\{-ur^2\} dr = 2\pi\lambda_{CM} . \\ \left(e^{-\frac{d^2 k u}{4}}\left[\frac{d^2 k}{8u} + \frac{1}{2u^2}\right]\right) \end{gathered} \tag{15}$$

Consequently, expression (12) can be simplified to the following expression:

$$\begin{gathered} E\left[\sum_{node \in cluster(j)} d^2_{toCH}\right] = 2\pi\lambda_{CM} . \\ \left(\frac{d^4}{64} + e^{-\frac{d^2 k u}{4}}\left[\frac{d^2 k}{8u} + \frac{1}{2u^2}\right]\right) \end{gathered} \tag{16}$$

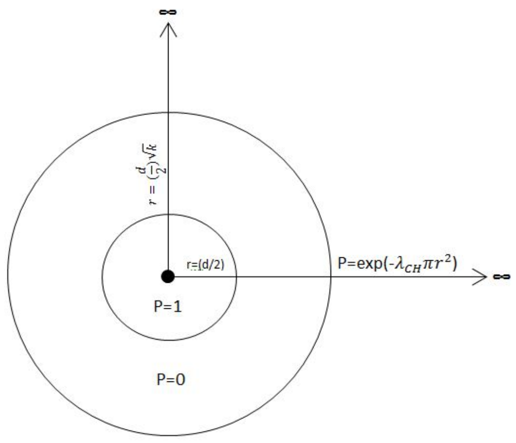


Fig. 5: An example of the first and second parts of cluster areas defined in the analytical model for ANCH.

The first and second parts of the network areas can be shown in Figure 5. In this figure, the inner circle shows the first part of each cluster in which $P\left\{(r, j) \in cluster(j)\right\} = 1$. The area between inner and outer circles demonstrates the first part of the other clusters in which $P\left\{(r, j) \in cluster(j)\right\} = 0$. The area beyond the outer circle shows the second part of the current cluster in which $P\left\{(r, j) \in cluster(j)\right\} = exp\left\{-\lambda_{CH}\pi r^2\right\}$.

The accuracy of the proposed analytical model for ANCH is evaluated in the next section.

V. Model Validation

The accuracy of the described analytical model has been verified by comparing it with simulation results. Extensive validation experiments have been performed for several combinations of cluster size, network dimension, different values of *closeness*, density of sensors in the network, and the number of messages which are sent from CMs to their CHs during the steady phase, called *MNumbers*. In order to select parameter a, different values including $a = 0.02, 0.05, 0.15, 0.25, ..., 0.75$ have been considered and the most effective value has been selected. Each simulation scenario is run for 100 different randomly generated topologies and the average results are presented. In our experiments, the sensors' inner computational procedures do not consume energy: all of their energy is used for message passing only. The energy model in all of our experiments is precisely the same as the one employed in [9].

As in the first experiment, the effects of varying the number of clusters on the accuracy of our proposed model is compared against the results obtained from simulation. The network area is considered to be 50×50 square metres when the base station is 100 metres away from the network's edge. Moreover, d = 15 metres and the initial energy of each node is 10 J. Finally, the number of clusters in this experiment varies from 4 to 15 clusters. The results are presented in Figures 6 and 7. In these figures, the horizontal axis shows the number of clusters where the vertical axis represents the total consumed energy. Figure 6 shows the accuracy of our model for three different networks with different numbers of nodes, N = 50, 100, and 200, when *MNumber* is considered to be 25. Also, Figure 7 shows the accuracy of our model for three different *MNumber*s, *MNumber* = 25, 50, and 100 messages per round, when the number of nodes is N = 100. 96.3% accuracy in Figure 6 and 97.1% in Figure 7 show that the simulation results closely match those predicted by the analytical model.

In the second experiment, we aim at observing the impact of network size on our analytical model. Different network dimensions from 10 to 100 metres are examined while the value of d is 30% of one dimension. Moreover, the initial energy of each node is 10 J and the number of clusters, k, is 5. These are depicted in Figures 8 and 9, highlighting that the proposed model on average presents an accuracy of 95.4% and 98.6%, respectively. Figure 8 shows the accuracy of our model for three different networks with different numbers of nodes, N = 50, 100, and 200, when *MNumber* is considered to be 25. Also, Figure 9 shows the accuracy of our model for three different *MNumber*s, *MNumber* = 25, 50, and 100 messages per round, when the number of nodes is N = 100.

In the third experiment, we aim at observing the impact of the *closeness* parameter, d, on our analytical model. Different *closeness* values from 5 to 25 metres are examined where the network area is considered to be 50×50 square metres and the base station is 100 metres away from the network's edge. Moreover, the initial energy of each node is 10 J and the number of clusters is 5. This is depicted in Figures 10 and 11, highlighting very close agreement between the model and the simulation in these figures, with 95.8% and 95.6% similarity on average. Figure 10 demonstrates the accuracy of the proposed model for three different networks with different numbers of nodes, N = 50, 100, and 200, when *MNumber* is considered to be 25. Also, Figure 11 shows the accuracy of our model for three different *MNumbers*, *MNumber* = 25, 50, and 100 messages per round, when the number of nodes is N = 100.

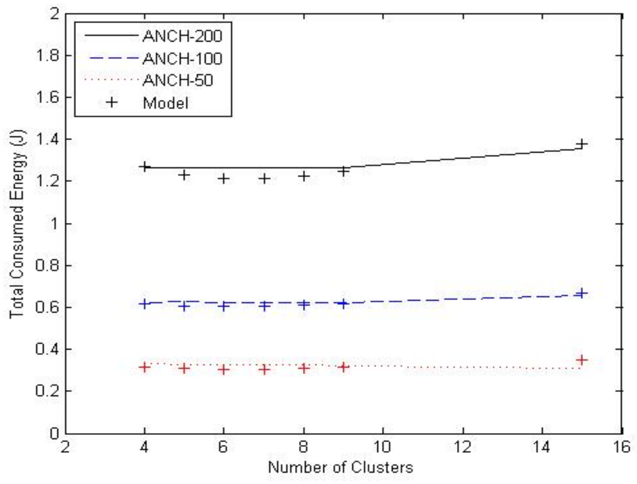


Fig. 6: Accuracy of the model comparing against simulation results varying the number of clusters for three networks with different numbers of nodes, N = 50, 100, and 200.

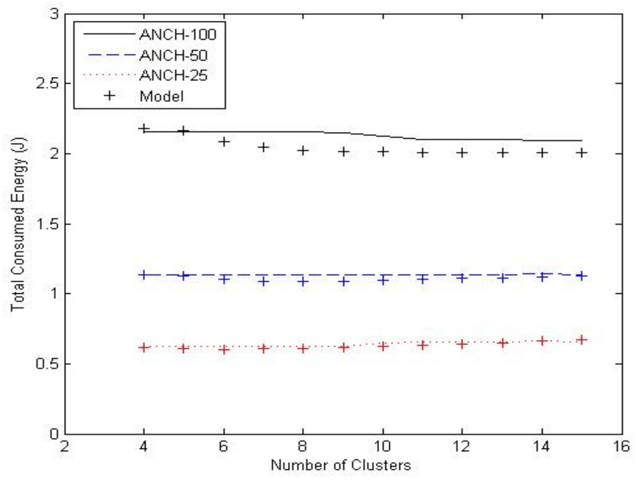


Fig. 7: Accuracy of the model comparing against simulation results for three values of *MNumber*, *MNumber* = 25, 50, and 100 messages per round.

In the fourth experiment, we aim at observing the impact of network density on our analytical model. In this experiment, different numbers of sensors, from 40 to 500, are examined.

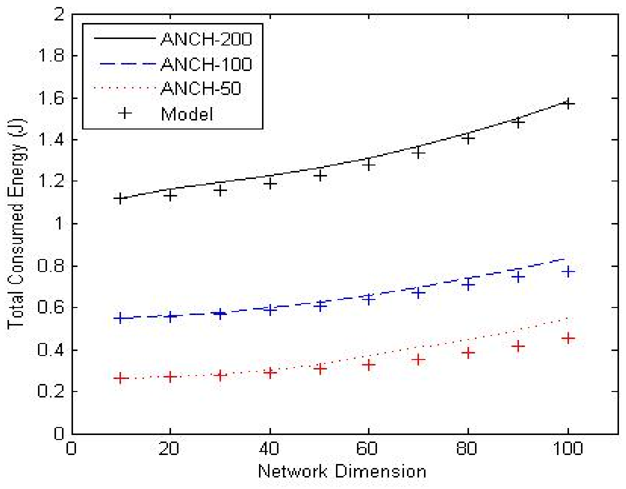


Fig. 8: Accuracy of the model comparing against simulation results varying the network dimension for three networks with different numbers of nodes, N = 50, 100, and 200.

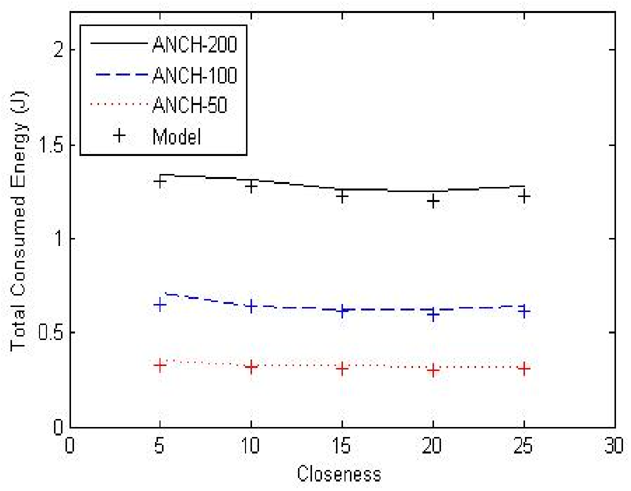


Fig. 10: Accuracy of the model comparing against simulation results varying parameter d for three networks with different numbers of nodes, N = 50, 100, and 200.

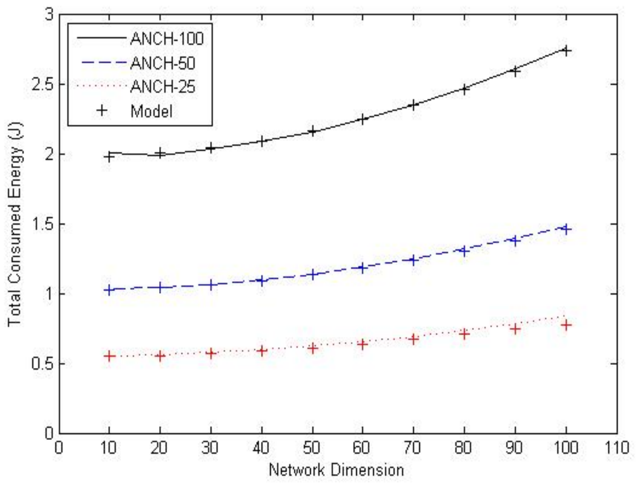


Fig. 9: Accuracy of the model comparing against simulation results for three values of *MNumber*, *MNumber* = 25, 50, and 100 messages per round.

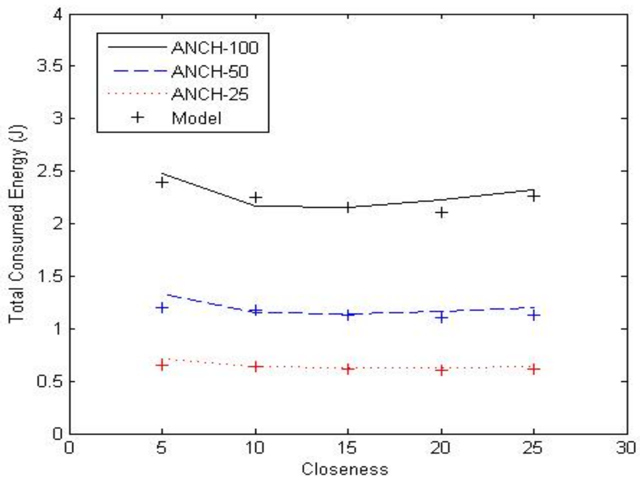


Fig. 11: Accuracy of the model comparing against simulation results for three values of *MNumber*, *MNumber* = 25, 50, and 100 messages per round.

Moreover, the network area is 50×50 square metres when the base station is 100 metres away from the network's edge, d = 15 metres, the initial energy of each node is 10 J, and the number of clusters is 5. The results are presented in Figure 12 for three different configurations, *MNumber* = 25, 50, and 100.

These results show a close agreement, an accuracy of 95.4% on average, between the proposed model and simulation results.

Finally, in the last experiment, we aim at observing the impact of steady phase duration on our analytical model by varying the number of *MNumber*s from 5 to 1000 messages per round. The network area is 50×50 square metres when the base station is 100 metres away from the network's edge, d = 15 metres, the initial energy of each node is 10 J, and the number of clusters is 5. In Figure 13, the comparison of the model and the simulation results for three different networks with N = 50, 100, and 200 nodes are presented, achieving 95.6% accuracy on average.

Overall, our extensive validation study shows the credible accuracy of our proposed analytical model in predicting the total energy spent by the ANCH algorithm.

Using the proposed model, a number of factors have been revealed. First, the energy consumed by the ANCH algorithm is almost insensitive to the optimum number of clusters, k, proposed by the LEACH algorithm. This is due to the important role of the add-on coefficient, a, in balancing the

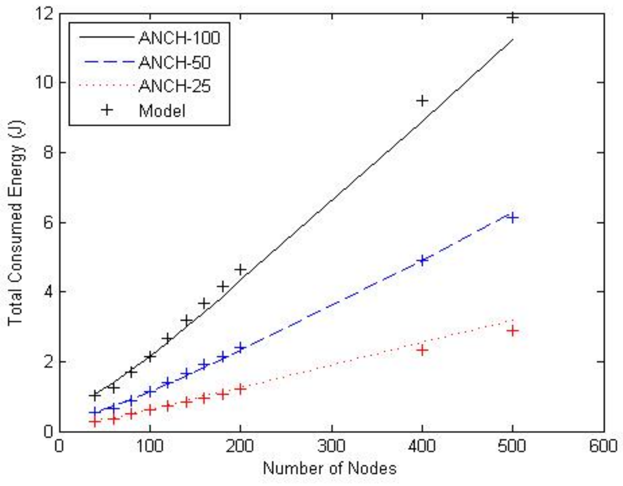


Fig. 12: Accuracy of the model comparing against simulation results for three values of *MNumber*, *MNumber* = 25, 50, and 100 messages per round.

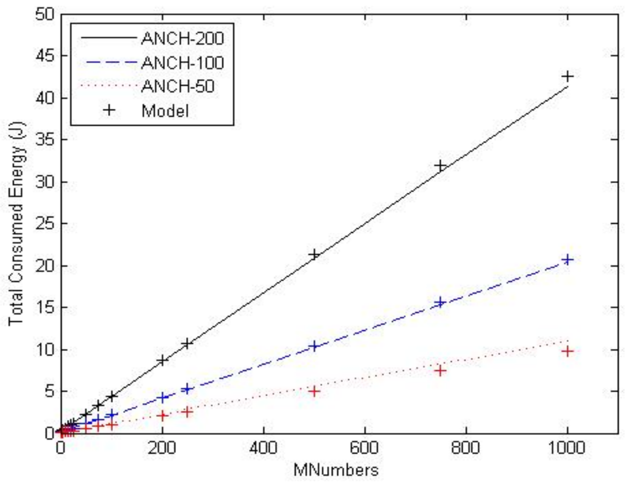


Fig. 13: Accuracy of the model comparing against simulation results for three networks with different numbers of nodes, N = 50, 100, and 200.

energy consumption of each cluster. By increasing the value of k, the optimum value of a is also increased to protect the network from forming a large number of clusters with smaller numbers of nodes in each cluster and hence to avoid wasting energy. Concomitantly, the optimum value of a is also decreased to block the negative effects of smaller numbers of clusters.

In the same way, the energy consumed by the ANCH algorithm is almost insensitive to the *closeness* parameter. This is again due to the balancing role of the add-on coefficient, a. By increasing the value of the *closeness* parameter, the optimum value of a is also increased to increase the number of potential CHs, avoiding smaller numbers of clusters. It also prevents the formation of large numbers of clusters when the *closeness* value is decreased.

VI. Conclusions and Future Work

Energy efficiency is essential in the design of practical WSNs. One of the most recent suggested distributed energy-efficient algorithms for them is the ANCH algorithm. ANCH extends the network lifetime by uniform distributing CHs throughout the network environment. In this manuscript, an analytical model for ANCH has been presented which demonstrates the impact of different factors in ANCH energy consumption and which can estimates its energy consumption in different situations. Our comprehensive research has shown a 95.4% to 98.6% accuracy in energy consumption estimation using the suggested analytical model compared to simulations. Also, the suggested analytical model has shown that the energy depletion of the ANCH algorithm is unresponsive to the *closeness* parameter and the number of clusters due to the adjusting role of the add-on coefficient in optimising the total energy depletion of clusters. In our future work, we are going to adapt the ANCH algorithm and its analytical model for mobile sensors and evaluate them using other simulation softwares.

References

[1] M. M. Zanjireh, H. Larijani, W. O. Popoola, and A. Shahrabi, "Analytical Modelling of ANCH Clustering Algorithm for WSNs," in *13 International Conference on Networks (ICN)*, (Nice, France), pp. 68–73, IARIA, February 2014.

[2] I. F. Akyildiz, S. Weilian, Y. Sankarasubramaniam, and E. Cayirci, "A Survey on Sensor Networks," *IEEE Communications Magazine*, vol. 40, no. 8, pp. 102–114, 2002.

[3] C. Jong-Moon, N. Yeonghun, l. P. Kyucheo, and J. C. Hyoung, "Exploration time reduction and sustainability enhancement of cooperative clustered multiple robot sensor networks," *IEEE Network*, vol. 26, no. 3, pp. 41–48, 2012.

[4] G. Liu, R. Tan, R. Zhou, G. Xing, W. Z. Song, and J. M. Lees, "Volcanic Earthquake Timing Using Wireless Sensor Networks," in *12th ACM/IEEE Conference on Information Processing in Sensor Networks (IPSN)*, pp. 91–102, ACM, 2013.

[5] C. Perera, A. Zaslavsky, P. Christen, and D. Georgakopoulos, "Sensing as a Service Model for Smart Cities Supported by Internet of Things," *Transactions on Emerging Telecommunications Technologies*, vol. 25, no. 1, pp. 81–93, 2014.

[6] M. M. Zanjireh, A. Kargarnejad, and M. Tayebi, "Virtual Enterprise Security: Importance, Challenges, and Solutions," *WSEAS Transactions on Information Science and Applications*, vol. 4, no. 4, pp. 879–884, 2007.

[7] K. Sohrabi, J. Gao, V. Ailawadhi, and G. J. Pottie, "Protocols for Self-Organization of a Wireless Sensor Network," *IEEE Personal Communications*, vol. 7, no. 5, pp. 16–27, 2000.

[8] C. Tselikis, S. Mitropoulos, N. Komninos, and C. Douligeris, "Degree-Based Clustering Algorithms for Wireless Ad Hoc Networks Under Attack," *IEEE Communications Letters*, vol. 16, no. 5, pp. 619–621, 2012.

[9] W. R. Heinzelman, A. Chandrakasan, and H. Balakrishnan, "Energy-Efficient Communication Protocol for Wireless Microsensor Networks," in *33rd Annual Hawaii International Conference on System Sciences*, 2000.

[10] S. Selvakennedy and S. Sinnappan, "An energy-efficient clustering algorithm for multihop data gathering in wireless sensor networks," *Journal of Computers*, vol. 1, no. 1, pp. 40–47, 2006.

[11] A. Maizate and N. El Kamoun, "Efficient Survivable Self-Organization for Prolonged Lifetime in Wireless Sensor Networks," *International Journal of Computer Applications*, vol. 58, no. 16, pp. 31–36, 2012.

[12] M. M. Zanjireh, A. Shahrabi, and H. Larijani, "ANCH: A New Clustering Algorithm for Wireless Sensor Networks," in *27th International Conference on Advanced Information Networking and Applications Workshops (WAINA)*, pp. 450–455, IEEE, 2013.

[13] I. Baidari, H. B. Walikar, and S. Shinde, "Clustering Wireless Sensor and Wireless Ad Hoc Networks using Dominating Tree Concepts," *International Journal of Computer Applications*, vol. 58, pp. 6–9, November 2012.

[14] P. Kumarawadu, D. J. Dechene, M. Luccini, and A. Sauer, "Algorithms for Node Clustering in Wireless Sensor Networks: A Survey," in *4th International Conference on Information and Automation For Sustainability (ICIAFS)*, pp. 295–300, 2008.

[15] A. Al Islam, C. S. Hyder, H. Kabir, and M. Naznin, "Finding the Optimal Percentage of Cluster Heads from a New and Complete Mathematical Model on LEACH," *Wireless Sensor Network*, vol. 2, no. 2, pp. 129–140, 2010.

[16] L. S. Bai, R. P. Dick, P. H. Chou, and P. A. Dinda, "Automated Construction of Fast and Accurate System-Level Models For Wireless Sensor Networks," in *Design, Automation Test in Europe Conference Exhibition (DATE)*, pp. 1–6, 2011.

[17] I. Beretta, F. Rincon, N. Khaled, P. R. Grassi, V. Rana, and D. Atienza, "Design Exploration of Energy-Performance Trade-Offs for Wireless Sensor Networks," in *49th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1043–1048, 2012.

[18] O. Younis and S. Fahmy, "HEED: A Hybrid, Energy-Efficient, Distributed Clustering Approach for Ad Hoc Sensor Networks," *IEEE Transactions on Mobile Computing*, vol. 3, no. 4, pp. 366–379, 2004.

[19] A. Wang, D. Yang, and D. Sun, "A clustering algorithm based on energy information and cluster heads expectation for wireless sensor networks," *Computers & Electrical Engineering*, vol. 38, no. 3, pp. 662–671, 2012.

[20] M. M. Zanjireh, H. Larijani, and W. O. Popoola, "Activity-aware Clustering Algorithm for Wireless Sensor Networks," in *9th IEEE/IET International Symposium on Communication Systems, Networks, and Digital Signal Procesing (CSNDSP)*, (Manchester, UK), pp. 132–137, July 2014.

[21] W. B. Heinzelman, A. P. Chandrakasan, and H. Balakrishnan, "An Application-Specific Protocol Architecture for Wireless Microsensor Networks," *IEEE Transactions on Wireless Communications*, vol. 1, no. 4, pp. 660–670, 2002.

[22] S. D. Muruganathan, D. C. F. Ma, R. I. Bhasin, and A. Fapojuwo, "A Centralized Energy-Efficient Routing Protocol for Wireless Sensor Networks," *IEEE Communications Magazine*, vol. 43, no. 3, pp. 8–13, 2005.

[23] N. Choon-Sung, J. Hee-Jin, and S. Dong-Ryeol, "The Adaptive Cluster Head Selection in Wireless Sensor Networks," in *IEEE International Workshop on Semantic Computing and Applications (IWSCA)*, pp. 147–149, 2008.

[24] M. J. Handy, M. Haase, and D. Timmermann, "Low Energy Adaptive Clustering Hierarchy with Deterministic Cluster-Head Selection," in *4th International Workshop on Mobile and Wireless Communications Network*, pp. 368–372, 2002.

[25] G. Smaragdakis, I. Matta, and A. Bestavros, "SEP: A Stable Election Protocol for clustered heterogeneous wireless sensor networks," tech. rep., Boston University Computer Science Department, 2004.

[26] I. Gupta, D. Riordan, and S. Sampalli, "Cluster-head Election using Fuzzy Logic for Wireless Sensor Networks," in *3rd Annual Research Conference on Communication Networks and Services*, pp. 255–260, 2005.

[27] G. Ran, H. Zhang, and S. Gong, "Improving on LEACH Protocol of Wireless Sensor Networks Using Fuzzy Logic," *Journal of Information & Computational Science*, vol. 7, no. 3, pp. 767–775, 2010.

[28] D. Karaboga, S. Okdem, and C. Ozturk, "Cluster based wireless sensor network routing using artificial bee colony algorithm," *Wireless Networks*, vol. 18, no. 7, pp. 847–860, 2012.

[29] A. A. Aziz, Y. A. Sekercioglu, P. Fitzpatrick, and M. Ivanovich, "A Survey on Distributed Topology Control Techniques for Extending the Lifetime of Battery Powered Wireless Sensor Networks," *IEEE Communications Surveys Tutorials*, vol. 15, no. 1, pp. 121–144, 2013.

[30] L. Chengfa, Y. Mao, C. Guihai, and W. Jie, "An Energy-Efficient Unequal Clustering Mechanism for Wireless Sensor Networks," in *IEEE International Conference on Mobile Adhoc and Sensor Systems*, pp. 598–604, 2005.

[31] Z. Qi, R. H. Jacobsen, and T. S. Toftegaard, "Bio-inspired Low-Complexity Clustering in Large-Scale Dense Wireless Sensor Networks," in *IEEE Global Communications Conference (GLOBECOM)*, December 2012.

[32] J. Gupchup, A. S. Terzis, R. Burns, and A. Szalay, "Model-based event detection in wireless sensor networks," *arXiv preprint arXiv:0901.3923*, 2009.

[33] S. Bandyopadhyay and E. J. Coyle, "An Energy Efficient Hierarchical Clustering Aalgorithm for Wireless Sensor Networks," in *22nd Annual Joint Conference of the IEEE Computer and Communications (INFOCOM)*, pp. 1713–1723, 2003.

[34] C. Jin-Chul and L. Chae-Woo, "Energy Modeling for the Cluster-Based Sensor Networks," in *6th IEEE International Conference on Computer and Information Technology (CIT)*, pp. 218–223, 2006.

[35] S. Foss and S. A. Zuyev, *On a certain segment process with Voronoi clustering*. France: Institut national de recherche en informatique et en automatique, 1993.

A Policy for Group Handover Attempts over Heterogeneous Networks

Nivia Cruz Quental and Paulo André da S. Gonçalves
Centro de Informática – CIn
Universidade Federal de Pernambuco – UFPE
Recife, Brazil
Email: ncq@cin.ufpe.br, pasg@cin.ufpe.br

***Abstract*—The proliferation of heterogeneous wireless networks and devices with multiple wireless link-layer technologies have called attention to the development of efficient vertical handover policies. Research on providing efficient vertical handover includes either the proposal of novel solutions or adaptations into existing horizontal handover schemes. In this work, we propose a policy for Group Vertical Handover (GVHO) attempts. We apply such a policy to an existing GVHO scheme, which handles vertical group handover based on a threshold that limits handover blocking probability. Performance is evaluated through simulation under several scenarios. To provide more realistic situations, we consider channel holding time in our studies. In addition, we study the fraction of blocked nodes and we vary threshold values for blocking probability. We compare our solution to that of the studied GVHO scheme. Results show that our solution reduces the handover latency and the fraction of blocked nodes while maintaining the handover blocking probability under a predefined threshold. In particular, latency is reduced from 11% to 51.5% in some of the scenarios studied.**

Keywords-GVHO; handover; policy of attempts.

I. Introduction

Load balancing and handover among different Radio Access Technologies (RATs) are the main concerns in Group Vertical Handover (GVHO) studies [1]-[3]. Research on GVHO covers simultaneously issues from Group Handover (GHO) [4]-[6] and Vertical Handover (VHO) [7]-[18].

GHO takes place when two or more Mobile Nodes (MNs) intend to request handover at the same time to the same base station. During GHO, MNs are not necessarily aware of the presence of each other. Thus, GHO procedures must carry out load balancing. To achieve this, criteria such as energy saving, available bandwidth, and type of service may be considered.

The continuity of telephone calls and streaming sessions over heterogeneous networks are covered by the VHO research field. Quality of Service (QoS) and the type of traffic may also define requirements for handover decisions apart from the underlying network technology available. IEEE 802.21 [19] is an example of effort to standardize VHO procedures and facilitate the proposal of new VHO solutions.

Providing support to GVHO has been motivated by the recent popularity of devices such as tablets and smartphones, which are capable of supporting multiple link-layer technologies and handling different kinds of traffic. Additionally, use cases involving users moving in trains and on buses are becoming more common and introduce new challenges. At the IP level, protocols like Proxy Mobile IPv6 [20] manage mobility sessions at the network layer. In this paper, we are particularly interested in the link layer handover.

Research on GVHO may involve the three main handover phases: discovery, decision, and execution [18]. The decision phase interests us the most, since the decision process in GVHO is still an open issue and it may impact the GVHO overall performance. Further, the decision algorithm itself must be associated to an optimized policy for GVHO attempts to guarantee better handover performance. Research on GVHO seeks to provide efficient decision-making techniques with their own policy for handover attempts. Some of them are based on centralized entities [21], distributed algorithms [22], random delays [2], reinforcement learning [23], game theory [2], and optimization problems [3]. We give special attention to Lee *et al.* [3], since it addresses the latency reduction while considering load balancing, support to legacy networks, and handover blocking probability. A reduced GVHO latency means less time spent in the GVHO operation. Load balancing is the consequence of a efficient resource management. Controlling the handover blocking probability means that the probability of the MN having its handover request denied by the target network is limited. Those issues are fundamental for advances in GVHO. The objective of Lee *et al.* [3] is to model GVHO decision as an optimization problem. Latency is minimized given the condition of maintaining the handover blocking probability under a predefined threshold. Although Lee *et al.* [3] present encouraging results, we find optimization opportunities in the policy for handover attempts.

In this paper, we propose a policy of attempts for GVHO. Our policy is based on exponential backoff and uses information from the GVHO scheme itself. We improve on previous experiments [1] by considering channel holding time in performance evaluations, which makes studied scenarios more realistic. In addition, we study the fraction of blocked nodes and several thresholds for the blocking probability. The proposed solution reduces average latency and the fraction of blocked nodes in comparison to results found in [3].

The remainder of paper is organized as follows: we present GVHO concepts in Section II. We present related work in Section III. We detail the GVHO scheme proposed in [3] in Section IV. We present the proposed policy for GVHO attempts in Section V. We present performance evaluation results in Section VI. Finally, we highlight our conclusions in Section VII.

II. Group Vertical Handover - GVHO

Recently, the concept of handover has evolved to take into account the continuity of communication sessions even among different RATs [24]. Technological evolution has allowed the rising of cheaper gadgets supplied with multiple network interfaces. The appearance of such gadgets, in turn, has encouraged new research in mobility management considering brand-new use-cases. Studies in the Group Vertical Handover (GVHO) area of interest aim at managing different connections taking place at the same time in public spaces with a diverse number of available technologies.

An example of GVHO scenario is illustrated in Figure 1. Suppose an open event, like a music festival where users desire to communicate with friends and transmit multimedia data. In this scenario, users are constantly changing their location. There may be several available RATs and dozens of devices in communication sessions simultaneously. If there are commercial agreements among the telecommunication carriers, it must be possible to maintain a communication session even if a group of users move from one network to another at the same time.

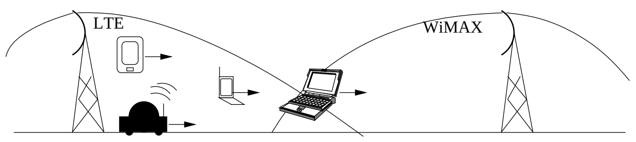


Figure 1. A GVHO scenario.

The integration between heterogeneous networks can be divided in two approaches: loose coupling and tight coupling [7]. In the loose approach, heterogeneous networks are integrated a the IP level, but they operate independently at the link layer. In this case, it is necessary to rely on a gateway to support authentication and accounting. Since handover is done at link layer, routing tables and authentication information must be updated. In order to accomplish this operation, mobility support is added to IPv4 and IPv6 protocols. The Internet Engineering Task Force (IETF) efforts in mobility management are concentrated at the IP level and layers above. The main protocols derived from that effort are Mobile IP [25], Mobile IPv6 Fast Handovers [26], PMIPv6 [20], and Fast Handovers for Proxy Mobile IPv6 (FPMIPv6) [27]. These standards allow the MN to maintain its initial IP address, even when it is out of its home network. The main advantage of loose coupling is the simple adaptation to legacy systems. However, handling handover only at the IP level may not entirely solve the problem of interruption of communication during this operation.

In the tight approach for heterogeneous networks coupling, the network entities of each technology must explicitly collaborate with each other. The authentication, communication management, and accounting are integrated. This approach requires more standardization effort than the loose approach. However, the handover management becomes more effective in terms of the number of lost packets.

For all layers of the protocol stack, the proposal of efficient and effective handover procedures for mobility management including handover decisions and optimal resource allocation is a critical need. In this paper, we are particularly interested in the efficiency of GVHO at link layer, when tight coupling takes place. In that context, proposals may involve the three handover phases [18]:

- *Discovery* - Service discovery and network information gathering. A specific criterion is adopted to determine if handover is necessary. According to [28], the gathered information may have a predetermined nature, like user policies and preferences, or time-varying nature, like signal-to-noise ratio, transmission rate, Point of Attachment (PoA) load, battery consumption, Received Signal Strength (RSS), RSS with threshold, RSS with hysteresis, etc.
- *Decision* - One network in a list of candidates is chosen, taking into consideration data collected in the earlier phase. Depending on the network technology, handover may be MN-initiated or network-initiated. The decision technique and the policy for handover attempts strongly impacts the resource management and the overall handover performance. Thus, the decision phase is the focus of this paper.
- *Execution* - Networks and MNs exchange control messages to make channel switching. This phase should minimize service interruption in order to appear imperceptible to the user. This phase is strongly media-dependent. The Hard Handover (HHO) implementation is mandatory for all technologies. HHO takes place when the MN disconnects from its original PoA before making the first contact with its target PoA. Since packages may be lost during that interval, optional Soft Handover (SHO) mechanisms are proposed. SHO mechanisms include Seamless Handover; Entry Before Break (EBB); Multicarrier Handover; Fast Base Station Switch (FBSS), or Fast Cell Selection (FCS); and Macro-Diversity Handover(MDHO) [12].

IEEE 802.21 standard [19], which describes the Media Independent Handover (MIH) can help determining the requirements for discovery and decision phases. MIH intends to be a common mean over the link layer in order to allow different RATs to communicate with each other during handover, abstracting implementation details. MIH is still a relatively new standard and, therefore, it faces challenges such as abstracting wireless technologies in a single interface, incorporation into existing handover schemes, security issues, power management, and storage issues at the information service.

Each RAT must provide its own implementation of MIH and must map the MIH messages to its media-dependent primitives. The main elements of MIH are:

- MIH Function (MIHF) - It detects changes in link layer, controls link state and provide neighborhood information;
- Service Access Points (SAPs) - It defines media-dependent/independent interfaces;

- MIH Users - Entities that make use of MIH services.

A handover scenario with MIH assumes the existence of an information service to help MNs to find neighborhood information. It avoids the MN to waste energy ant time making scanning operations by itself. There are plenty of studies on handover performance adopting MIH as an auxiliary tool for discovery and decision processes [3][29]-[32].

There may be many different decision criteria for GVHO such as available bandwidth, expected QoS, or battery consumption. The type of service (voice or data) is a determinant factor for choosing the most suitable criterion for GVHO decision. Decisions made without network analysis and without considering the MNs in the neighborhood may bring disastrous performance results. Wrong handover decisions may cause MNs to choose the same PoA, overloading it, or to choose an inadequate network for the application in use. The main handover decision approaches found in GVHO research include:

- *Centralized entities* [21][33] - A relay station handles GVHO management, removing complexity from MNs. This approach also reduces the uncertainty level and ensures better performance than decentralized approaches. The main drawback is the lower fault tolerance. Figure 2 illustrates an architecture based on central entity. If the entity suffers a failure, the mobility management would be damaged.

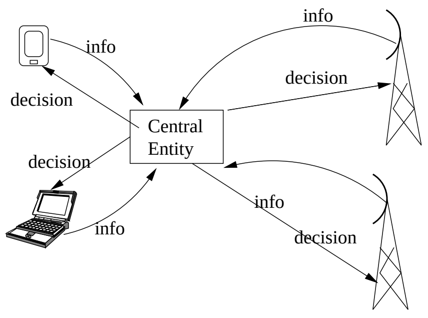


Figure 2. Central-entity-based approach.

- *Distributed algorithms* [22] - The decision algorithm makes use of well-known parallelism and synchronization techniques. Distributed algorithms are usually simple to understand. Figure 3 shows an example of distributed approach. The architecture is fault-tolerant, however, the algorithms are not built to adapt themselves to new scenarios.
- *Random delays* [2] - MNs attempt to handover after a random delay. This procedure minimizes simultaneous handover attempts and is considered a subtype of the distributed algorithm approach. In Figure 4, we present an example of handover that happens in different instants of time for each MN. This approach avoids collision among MNs, distributing handover requests over time.
- *Reinforcement learning* [23] - It employs Artificial Intelligence (AI) techniques to make MNs learn about their surrounding environment as they make handover

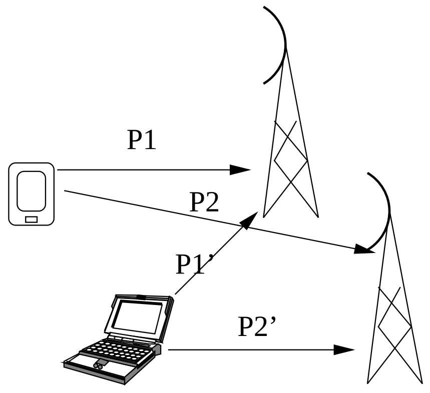


Figure 3. Distributed-algorithm-based approach.

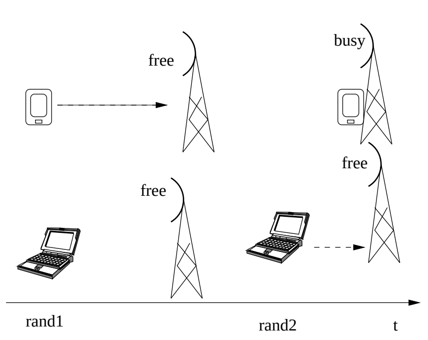


Figure 4. Random-delay-based approach.

attempts. This approach does not require message exchange among users; they use the information received from other entities over time. Figure 5 presents this interaction. However, learning algorithms may cause performance issues due to the complex processing.

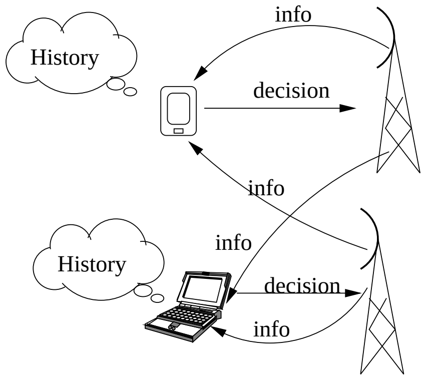


Figure 5. Reinforcement-learning-based approach.

- *Game theory* [2][23] - This approach maps handover scenarios in cooperative or non-cooperative games in which MNs are players interested in getting the best payoff as possible, shown in Figure 6. The payoff may be a larger bandwidth, energy saving, or better security. Nash equilibrium is the desired stable state in which all MNs do not have anymore strategies to obtain better payoffs. The main advantage of this approach is the almost perfect match between a GVHO scenario and the Game Theory competitive models. On the other hand, it is not always possible to model additional parameters.

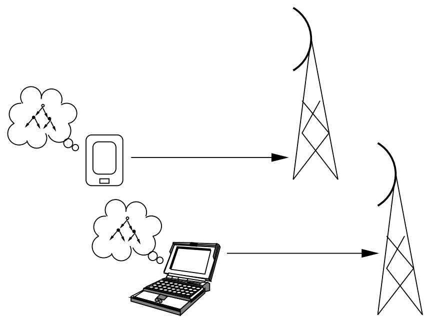


Figure 6. Game-theory-based approach.

- *Mathematical optimization problems* [3] - Mathematical equations are used to describe the handover decision under predetermined conditions. Figure 7 illustrates that approach. The optimization problem is solved by finding the ideal value for the equation variables. This approach requires a more complex modeling and is more flexible than Game Theory-based models.

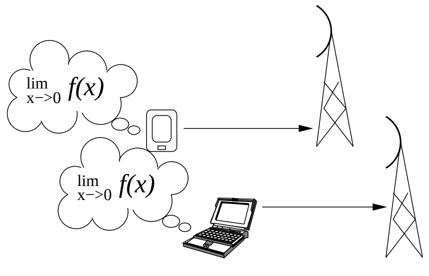


Figure 7. Optimization-problem-based approach.

For any GVHO approach, the MN or the serving PoA may determine if it is possible to request handover in a certain time, or if it is preferable to postpone it, given the network conditions. Policies for handover attempts can influence handover performance, for better or for worse, depending upon the adopted solution.

III. RELATED WORK

In this section, we present a critical analysis of recent research related to GHO, VHO, and GVHO.

A. Group Handover

Chowdhury *et al.* [5] propose a resource management scheme using a dynamic bandwidth reservation policy in mobile femtocellular network deployment. The scheme aims at the vehicular scenario and uses the proximity of new stations and required QoS as information to allocate the corresponding bandwidth only when necessary. Simulation results show a reduction in the handover call drop probability and maintains bandwidth utilization when compared to schemes without QoS criteria and without priority.

Jeong *et al.* [4] propose a specific handover scheme for the IEEE 802.16e standard. It consists on reducing the number of packets necessary to accomplish handover, by means of a group-based channel scan. The MNs form groups and inside each group there is a handover schedule for the MNs. Computer simulations and Markov models were used to compare the scheme performance with the existing scheme in the IEEE 802.16e standard. The authors observed that the proposed scheme reduces blocking probability.

Fu *et al.* [6] highlight a group-based authentication scheme for WiMAX networks. That scheme consists on the PoA sending security context to a group of MNs if a member of this group requests handover. The main objective is to reduce handover latency while maintaining privacy preservation. The metrics evaluated are latency, communication overhead and computation cost. Simulation results show that the two earlier metrics are reduced, however, the latter is increased.

B. Vertical Handover

The state-of-art in VHO schemes can be found in [12]. The authors give more highlights on IEEE 802.16m and 3GPP LTE-Advanced technologies. According to the authors, IEEE 802.16m offers enhancements to link layer performance, such as multicarrier handover, in comparison to IEEE 802.16 legacy. The paper also presents the supported handover procedures besides *Hard Handover*, such as *Seamless Handover* and *Entry Before Break*.

In [7], Park *et al.* propose integration between WiMAX and cdma2000 networks. Their approach takes elements from tight coupling and loose coupling, introducing new messages in link layer and establishing tunnels in the IP layer. Simulations with OPNET measured delay in function of the elapsed time and the speed of nodes. Packet loss ratio is also measured in function of the elapsed time. According to the authors, those metrics are reduced in comparison to a loosely-coupled scheme.

Kim *et al.* [8] present a proposal for the *Hierarchical Mobile IPv6* (HMIPv6). They propose to execute IP-level handover and link-layer handover simultaneously, in order to reduce total time. Results show the reduction of latency and package loss in a intra-domain scenario in comparison to HMIPv6.

Shen *et al.* [9] propose a cost-function-based network selection. The authors consider available bandwidth information, traffic load and RSS. Simulation results show the scheme behavior in different scenarios. The authors conclude that the proposed scheme affects several system parameters, which need to be handled carefully.

Stevens-Navarro *et al.* [10] uses a Markov decision process for VHO having the maximization of the total expected reward per connection as objective. Performance evaluation considers voice and data applications. Numerical results show that the proposed algorithm performs better than the simple-additive-weighting algorithm.

Yeh *et al.* [11] propose the *Fast Intra-Network and Cross-layer Handover* (FINCH). It is a complementary mechanism to Mobile IPv4 for intra-domain mobility management. The main objective is to reduce latency at the IP layer. FINCH uses cross-layering techniques, which allows a more efficient localization and path optimization. The authors use numerical simulation to compare FINCH to Mobile IP, Fast Mobile IP, HMIP, Cellular IP, and HAWAII. The authors observe that FINCH reduces location cost and overall latency.

Gondi *et al.* [13] propose to use network context information during handover. These information include location, required bandwidth, battery status, available network interfaces, and authentication key. The authors give special attention to security in VHO. Experiments are run in a testbed to demonstrate how the proposal can be deployed.

Choi *et al.* [14] propose a new metric for VHO decision: *Interference to other Interferences-plus-Noise Ratio* (IINR). The main objective is to enhance throughput by analyzing the interference among cells in a cooperative fashion. The MN only handover to another cell if there are possibility of throughput gains. The authors use simulation to prove that the proposed scheme increases throughput at the scenarios studied in comparison to schemes that use *Signal to Interference-plus-Noise Ratios* (SINRs) as decision metric.

Koh *et al.* [15] study the fast handover in wireless multicast networks. According to the authors, the message calls to the IGMP protocol can be optimized when introducing *Multicast Handover Agents* (MHAs) at the base stations. Numerical simulations show a reduction of delay in the scenarios studied.

Kim *et al.* [16] propose a common link layer for 3G, WiMAX, and WiBRO networks. Additionally, three decision schemes are presented based on available bandwidth and cost employing neural networks. Simulations measure throughput, cost, and handover success rate and show better results than RSSI-based schemes.

The work in [17] concerns with the TCP throughput during handover in a FPMIPv6 network. The solution includes MIH to make QoS negotiations, preregister, and pre-authentication. Simulations with OPNET show that the proposal reduces TCP overhead in comparison to FPMIPv6.

Zekri *et al.* [18] present a survey on VHO solutions. It highlights the main technical challenges in heterogeneous wireless networks underlying seamless vertical handover. The authors also presents the standards involved and present comprehensively the mobility management process.

C. Group Vertical Handover

In [21], a relay station is used as a centralized entity to coordinate GVHO. The scenario studied is the movement of users in a train. Handover blocking and interruption probabilities are evaluated with the increase of the calls-per-minute ratio. The evaluation compares schemes with and without the relay station. The authors conclude that the proposed scheme reduces handover blocking and interruption probabilities. In this case, the relay station is responsible for executing the policy for handover attempts. The solution has limitations if co-existence with legacy systems is needed. This is due to the need of introducing a new infrastructure with special requirements.

Ning *et al.* [33] propose that a network entity called *Radio Resource Management Center* (RRMC) is responsible for collecting data from the nodes and network candidates. Thus, the RRMC decides which group of nodes may handover to a given network. The decision is based on Fuzzy Clustering, which is used to group nodes with similar characteristics. The policy for handover attempts is totally controlled by that entity. Results show that the solution reduces the blocking probability in comparison to [3], a decentralized scheme. However, it does not give results for the latency and does not make comparisons to another centralized scheme.

Cai *et al.* propose three decentralized algorithms for GVHO in [2]. The first is a Nash equilibrium-based algorithm where the policy for handover attempts is based on the game strategy of each player. The second algorithm adopts random delays, thus using a simpler policy for handover attempts. The third algorithm is a more refined version of the previous one. It considers latency as a basis for delay calculations. Performance evaluations show that latency values under the three algorithms are similar. Handover blocking probability is not considered.

Niyato *et al.* propose a model for network selection that is based on evolutionary games [23]. The model consider two approaches: a central entity-based approach and a decentralized-based approach that uses a reinforcement learning model. In the first approach, the central entity controls handover attempts. In the second approach, MNs are allowed to infer the best period of time to request a handover. The fraction of MNs choosing the same PoA is the load-balancing metric adopted. They conclude that each approach has its advantages in accordance with the scenario. One drawback is not evaluating the impact of the approaches on latency.

Lei *et al.* [22] present three GVHO schemes. The first scheme schedules simultaneous attempts to random time periods. In the second scheme, MNs select PoAs using a predefined probability as a base. In this case, the policy of handover attempts consists in an immediate attempt. The last scheme requires the network to be responsible for the handover decision. Results show that the last approach is more efficient. However, it may be difficult to adapt it to legacy systems.

Lee *et al.* [3] propose a GVHO scheme, which is based on the solution of an optimization problem. The MN is responsible for the handover decision. The main objective is to minimize latency while limiting the handover blocking probability. Some factors make the scheme in [3] more promising than the other researches:

- it does not require the presence of a relay station.
- it may work together with legacy systems.
- it considers two of the main GVHO metrics: load balancing and latency.

We detail such scheme in Section IV.

IV. Reference GVHO scheme

Lee *et al.* [3] propose an optimization for the total handover latency L, considering the handover blocking probability as follows:

$$\begin{aligned}&\texttt{Minimize}\ L\\&\texttt{Subject to}\ P_{HoBlock}(t) \leq P_{HoBlockThreshold}\ ,\end{aligned}$$

where $P_{HoBlock}(t)$ is the handover blocking probability in a time t and $P_{HoBlockThreshold}$ is the maximum acceptable value for the handover blocking probability. Latency is calculated as follows:

$$L = N_{HO}.\Delta t, \tag{1}$$

where N_{HO} is the total number of attempts until the MN requests the handover; Δt is the period of time between consecutive attempts. If the MN decides to request in the first attempt, total latency would be Δt. This is because in [3], execution time is also equal to Δt.

Equation (2) presents the calculation of $P_{HoBlock}(t)$. The value of $P_{HoBlock}(t)$ is dependent on the number of candidate networks, their available bandwidth, and the number of participating MNs in GVHO. In [3], it is considered that these values can be obtained by using IEEE 802.21 MIH (*Media Independent Handover*) queries and *ad hoc* communication.

$$P_{HoBlock}(t) = \sum_{k=1}^{K} \sum_{i=C_k(t)}^{M-1} \frac{(i+1-C_k(t)).(M-1)!}{(i+1)!.(M-1-i)!} \times ((P_{sel}^k)^{i+1}.(1-P_{sel}^k)^{M-1-i}), \tag{2}$$

Where:

- M represents the number of participating MNs.
- K represents the number of candidate networks with overlapping areas.
- $C_k(t)$ is the available bandwidth in a time t for a network$_k$. The model considers that the available bandwidth is represented by an integer value. Each MN requires one unity for handover;
- P_{sel}^k : The probability of selecting network$_k$.

The Karush-Kuhn-Tucker (KKT) condition is used in optimization problems and it can be applied to (2) to determine the P_{sel}^k value. However, P_{sel}^k can be obtained by using (3), which is simpler than using KKT and induces minor changes in results.

$$P_{sel}^k(t) = C_k(t)/\sum_{k=1}^{K} C_k(t). \tag{3}$$

Now, we can find the $M_{optimal}(t)$ value that ensures the optimization problem condition. This value can be found by setting it initially to one, then increasing it by one unit while the $P_{HoBlock}(t)$ value is still less than or equal to $P_{HoBlockThreshold}$. This procedure is described in Algorithm 1.

Algorithm 1: Find $M_{optimal}$ value

```
$M_{optimal}$ = 0 ;
repeat
    p = Equation (2) ;
    $M_{optimal}$ = $M_{optimal}$ +1 ;
until $p \leq P_{HoBlockThreshold}$;
```

The probability $P_{HO}(t)$ with which a MN can request handover is given by:

$$P_{HO}(t) = M_{optimal}(t)/M. \tag{4}$$

If the MN decides not to request the handover immediately, a new attempt will be made after a constant time interval. The MN requires the number of attempts necessary to have a well-succeeded handover with blocking probability less than or equal to $P_{HoBlockThreshold}$. Algorithm 2 summarizes this process and can also be found in [3].

Algorithm 2: Reference GVHO scheme

```
L = 0;
c_atts = 1;
Mtotal = number of GVHO participants;
Mremaining = Mtotal;
while Mremaining $\leq$ 0 do
    find Moptimal in function of (2);
    calculate PHO;
    if decision(PHO) then
        choose networkk depending on $P_{sel}^k$;
        NHO = c_atts;
        break ;
    else
        L += t_atts(c_atts);
        c_atts++;
    end
    Mremaining = Mremaining - Moptimal
end
L += LHOexec;
```

Where:

- `Mtotal` is the total number of MNs in GVHO.
- `Mremaining` is a counter that checks for the end of algorithm.
- `decision()` is a function that returns `true` with probability $P_{HO}(t)$.
- `LHOexec` is the handover execution time. It is equal to Δt.
- `t_atts()` is a function to calculate the period of time between consecutive attempts. In [3], the return value of this function is always Δt.
- `c_atts` counts the number of attempts. When `decision()` is `true` in the first attempt, the total execution latency is `LHOexec`.

Function `t_atts()` characterizes the policy for handover attempts. In this case, it is a function that returns a constant value and it is equals to the execution latency L_{HOexec}.

V. The Proposed Policy for GVHO Attempts

Despite of presenting a promising GVHO scheme, the work in [3] lacks a good policy for handover attempts. It is based on a constant delay, which causes a negative impact on the overall GVHO performance as the number of MNs increases. In this section, we present a policy for GVHO attempts that aims at providing reduced handover latency for GVHO schemes like the one proposed in [3]. At the same time, we intend to reduce the latency and the number of blocked nodes, maintaining the blocking probability premise.

In order to enhance performance results, we propose to modify the `t_atts()` function in Algorithm 2. Our proposed solution is exponential backoff-based. It depends upon the `c_atts` counter and the duration of a reference slot time. It is a particular case of random delay. Exponential backoff algorithms have the particularity of keeping the probability of collision and the probability of transmission stable as the number of nodes which are sharing a medium grows [34]. Although our solution is motivated by the performance issues in [3], it is generic enough to be applied in other schemes. Equation (5) shows our modified version of `t_atts()`:

$$t\_atts(c\_atts) = \begin{cases} random[0..2^{c\_atts} - 1]\ .\ timeSlot\ , \\ \quad if\ c\_atts \leq LimBackFactor \\ random[0..2^{LimBackFactor} - 1]\ .\ timeSlot\ , \\ \quad otherwise \end{cases} \tag{5}$$

where $random$ picks a uniformly distributed number over the given interval; $LimBackFactor$ is the number of attempts that limits the range of values for $random$; and $timeSlot$ is the duration of a reference time slot, which depends on the target network. This information is obtained via MIH.

Total latency depends directly on the number of attempts, which varies with the return of `decision()`. The exponential backoff approach in `t_atts()` gives to the MN an opportunity for a new handover attempt after a time interval shorter than Δt, or even immediately. When the MN chooses not to request handover, other MNs may request it, reducing concurrency during the next attempts. Thus, MNs finish their handover sooner, decreasing total latency. Additionally, the number of nodes that have their handover blocked also decreases, making the handover more effective.

In [3], the return value of `t_atts()` is constant and equals to the execution latency L_{HOexec}. In that case, latency always grows by a constant factor. It causes a negative effect in the overall handover performance as the number of MN grows, as shown in [3]. Figure 8 presents the behavior of `t_atts()` in function of the number of attempts. We observe that the interval between attempts in (5) is always smaller than the approach in [3]. Analysis of the effect of the proposal in the overall performance is presented in the next section.

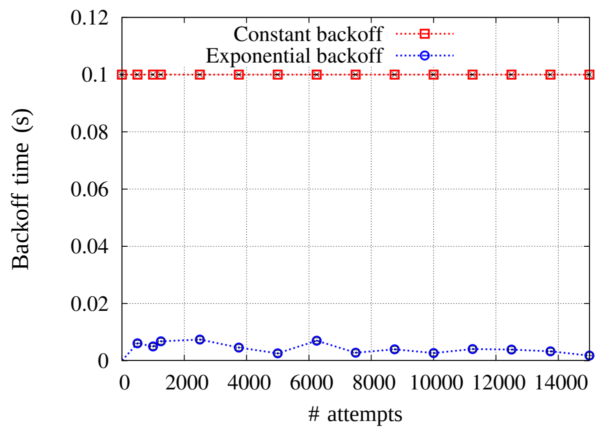


Figure 8. Comparison between the different implementations for the `t_atts()` function.

VI. Performance Evaluation and Comparison

In this section, we extend the experiments made in [1] introducing new parameter values and simulation conditions. The metrics evaluated in this paper are latency and handover blocking probability, as in [3] and [1], and additionally, the fraction of blocked MNs. The fraction of blocked MNs measures the fraction of nodes that decided to request handover and, for lack of bandwidth, had their request denied by the destination network. This metric helps us to evaluate the effectiveness of the GVHO scheme. All metrics are plotted in function of the number of MNs.

The majority of the parameters also follows the work in [3] and [1]. The value of Δt is set to 0.1s. We study scenarios with different values for $P_{HoBlockThreshold}$: 0.01, which is the recommended value according to Telecordia [35]; 0.02, which is a typical value [36][37]; and 0.05, in order to observe the effects of a less conservative parameter. The thresholds of 0.02 and 0.05 has been addressed in the former experiments [3][1] and the threshold of 0.01 is introduced in this paper. The number of MNs varies from 20 to 100, as in [1]. It differs from Lee *et al.* [3], where this number varies from 20 to 65.

In this paper, we introduce the Channel Holding Time (CHT) in the simulations. CHT is the time elapsed while a mobile node occupies a channel in a cell due to new connections or handover in an ongoing call [38]. In [1] and [3] is considered that all nodes leave network as soon as they handover to it. The introduction of the CHT factor give us a more realistic environment for analysis. The CHT modeling usually depends on the call holding time, the cell dimensions, cell residence time, resource allocation strategy, and the network architecture [39]. However, studies has shown that CHT can be approximated to a random variable with exponential distribution [38][39]. We consider 60s as the mean CHT. In other words, in our simulation, a set of nodes arrive, make handover attempts according to the policy adopted ; then, each one remain consuming a unit of bandwidth resource by a time defined by a random variable exponentially distributed with mean 60s.

We maintain the characterization of heterogeneity as the use of different available bandwidths to be compliant with the modeling presented in [3]. The number of available PoAs is 5, considering the following scenarios:

Scenario 1 - All PoAs have 20 bandwidth units.

Scenario 2 - Two PoAs have 15, two PoAs have 17, and one PoA has 20 bandwidth units, respectively.

Scenario 2 is only used in [3] for validating their simulator and in a situation of co-existing individual handover, which is out of the scope of this paper. Nevertheless, we include Scenario 2 in our evaluations. The $FatorLimBack$ parameter is set to 10. This value is based on preliminary experiments. We consider that MNs are switching from an arbitrary network to an IEEE 802.11 area. The parameter $timeSlot$ is set to $9.10^{-6}s$, which is equivalent to the SIFS time slot in IEEE 802.11 standard.

We have implemented the reference scheme and our solution in a discrete-event simulator, which was written in C++. Figure 9 illustrates how our scheduler operates in a given state, when *node 3* has its decision made. In this example, we have a queue of events, ordered by the scheduled time. Initially, we have *n* events of MNs trying to handover at the same time, in time *t = 0*. Then, each event is dequeued and processed according to the Algorithm 2. In this case, *node 3* decided to postpone handover request to time *t3*, enqueuing the corresponding event. In that state, *node 1* already had its event in *t = 0* processed and the decision to retry handover at time *t1* have been put at the queue. There is also another event for *node 2*, which decided to execute handover at time *t2*. Thus we can simulate parallelism in events, since bandwidth allocation will only happen in another event, when the node will in fact execute handover.

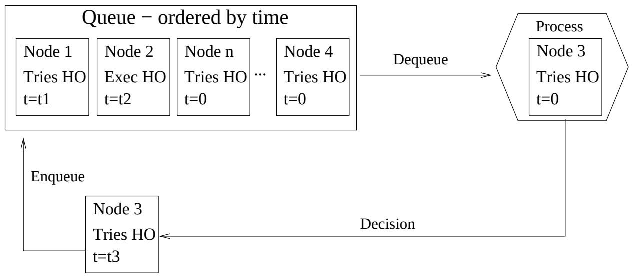


Figure 9. Example of scheduler instance in our discrete-event simulator.

The implementation of the reference scheme in our simulator was validated by the authors of [3]. We consider a group of MNs simultaneously entering a new coverage area and starting handover procedures defined by the GVHO scheme studied.

We represent confidence intervals with 99% of confidence level. Confidence intervals appear imperceptible in Figures 10-14. It is important to point out that we are not interested in evaluating the decision algorithm itself, but the impact of our policy for GVHO attempts on performance.

A. Results for Scenario 1

Figure 10 shows results for handover blocking probability under Scenario 1. The probability increases as the number of MNs grows from 55 for threshold 0.01, from 60 for threshold 0.02, and from 70 MNs for threshold 0.05. Thereafter, the curves are stable. This happens because blocking probability is getting closer to the threshold defined in the optimization problem. Since blocking probability is directly related to the cell utilization [40], it is necessary to limit the number of MNs entering a new cell at the same time in order to maintain the blocking probability under the threshold. When the blocking probability reaches the threshold, the value of $M_{optimal}(t)$ that is calculated in function of (2) can not increase anymore. This leads the remaining MNs to wait for another handover attempt. Thus, the stabilization of the blocking probability curve as the number of MN grows always implies the increase of the average latency. It is important to notice that the curves with and without our solution are similar because the optimization problem conditions are still the same. It means that the application of the proposed solution does not cause damages to the handover blocking probability, despite of the shorter time between attempts.

Figure 11 shows results for the fraction of blocked nodes under Scenario 1. We can observe that, as the number of

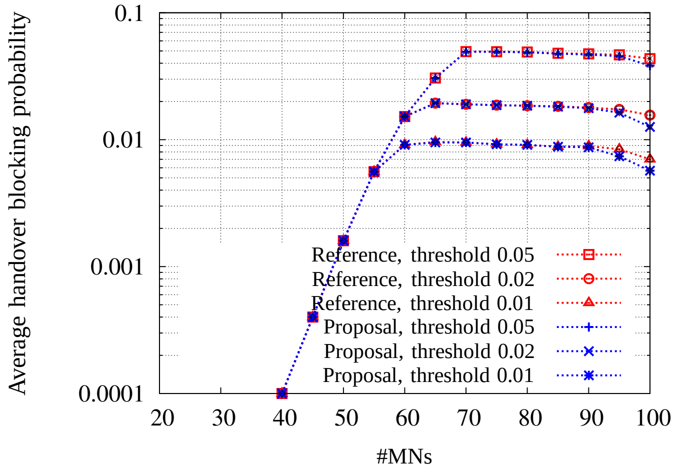


Figure 10. Handover blocking probability *versus* the number of MNs in Scenario 1.

MNs approximates to 100, the number of blocked MNs in the scheme in [3] is greater than the value found in the proposed solution. This is reflected in the blocking probability graph in Figure 10. For all thresholds, the blocking probability is slightly smaller when the number of nodes is between 95 and 100. It is due to the random nature of the attempts, which avoids collision among MNs.

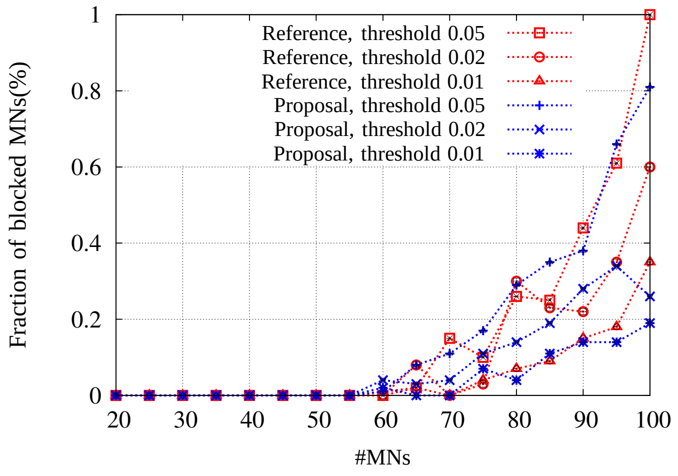


Figure 11. Fraction of blocked nodes *versus* the number of MNs in Scenario 1.

Figure 12 shows results for latency in Scenario 1. With respect to the scheme in [3], we can observe that latency starts growing from 55 MNs for threshold 0.01. For the threshold of 0.02, values start to grow at 60 MNs. Values in that curve are greater than those for threshold 0.05, which starts growing from 70 MNs. As we have stated before, the stabilization of the blocking probability curve observed in Figure 10 implies the increase of the average latency. Also, there is a greater number of handover attempts when we use a lower threshold. It tends to make MNs wait for more time with thresholds 0.01 and 0.02 than those using threshold 0.05. The lower the threshold is, the more conservative is the scheme and the greater is the average latency. We can also observe in Figure 12 the impact of the proposed solution on the latency curve. The curve is much smoother than the curve that does not adopt the solution.

For threshold 0.05, the latency is 11% smaller in the case of 80 MNs and 38% smaller for 100 MNs. For threshold 0.02, latency is 18% smaller for 80 MNs and 50% smaller for 100 MNs. Finally, for the threshold of 0.01, we observe a reduction of 22.5% for 80 MNs and 58% for 100 MNs.

The latency reduction is due to the proposed solution, which makes the delay between attempts more flexible. The exponential backoff also brought randomization to the scheme allowing MNs to try handover again sooner and in different periods of time, eventually reducing the total number of attempts.

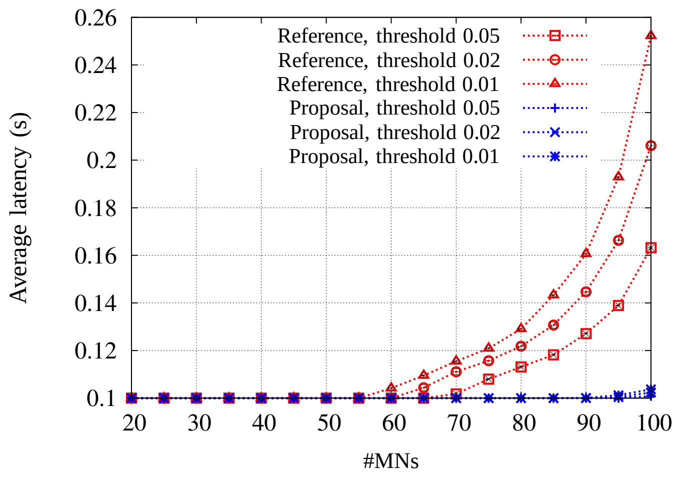


Figure 12. Latency *versus* the number of MNs in Scenario 1.

B. Results for Scenario 2

Figure 13 presents results for the Scenario 2. This figure presents similarities with Figure 10 but the curves stop growing sooner: from 50 MNs for the thresholds 0.01 and 0.02, and from 55 MNs for the threshold 0.05. This anticipation is due to the shorter total available bandwidth in the scenario studied. Thus, handover blocking probability increases faster, but it also gets stable in accordance with the established threshold.

However, we observe that the blocking probability starts to reduce again, from 85MNs. The explanation for this phenomenon is found in Figure 14. Figure 14 presents the fraction of blocked MNs for the Scenario 2. It is important to notice the expressive increase of the number of blocked nodes in the scheme in [3], which causes some nodes leave the concurrency because they were blocked. Thus, for the remaining nodes the blocking probability gets smaller. A similar phenomenon happens to the proposed solution, however, the number of blocked nodes is smaller, because the randomization of attempts makes handover requests less risky. We observe this behavior in all thresholds.

Figure 15 shows results for latency in Scenario 2. As in Scenario 1, the curves for thresholds 0.01 and 0.02 have greater latency values than the one with threshold 0.05. In [3], latency starts growing from 60 MNs for threshold 0.01, from

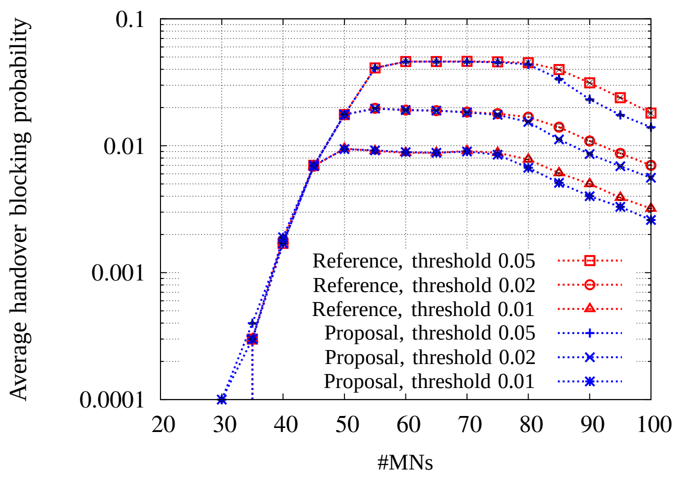


Figure 13. Handover blocking probability *versus* the number of MNs in Scenario 2.

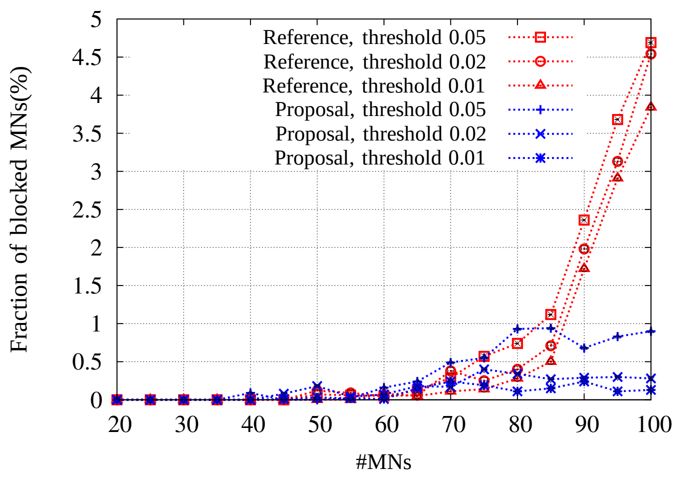


Figure 14. Fraction of blocked nodes *versus* the number of MNs in Scenario 2.

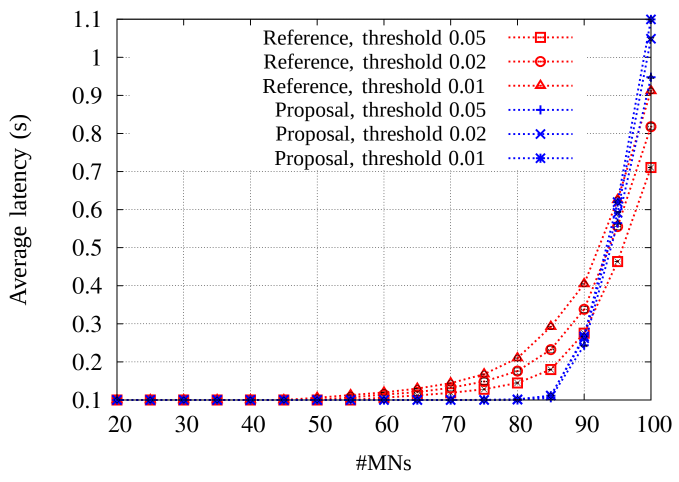


Figure 15. Latency *versus* the number of MNs in Scenario 2.

65 MNs for threshold 0.02, and from 70 MNs for threshold 0.05. Greater latency values are expected because the total available bandwidth is shorter than in Scenario 1.

Figure 15 also shows that for the threshold 0.05, latency has a reduction of 30% for 80 MNs. For the threshold 0.02, we observe a reduction of 42% for 80 MNs. In the threshold of 0.02, the latency is 51.5% smaller for 80 MNs. We also notice that from 95 MNs, our solution presents a greater latency than that one in [3]. The latency for 100MNs with our solution is 25%, 22%, and 17% greater than the scheme in [3] with the thresholds of 0.05, 0.02, and 0.01, respectively. It happens because, since our solution has a smaller percentage of blocked nodes, as shown in Figure 14. The remaining nodes, instead of being blocked as in [3], wait for more time for a handover opportunity and consequently, they increase the average latency.

VII. Conclusion and Future Work

In this paper, we have proposed a policy for GVHO attempts. Our solution uses exponential backoff in order to allow a better distribution of handover attempts over time. Performance evaluations have shown that our proposal makes it possible to reduce handover latency and the percentage of blocked nodes during handover. In particular, results have shown that latency was reduced up to 51.5% in accordance with the scenarios evaluated. Our future efforts will focus on including MIH queries in the solution design and including the information gathering phase in performance evaluation. Although this solution is well-suited to resolve performance issues in the scheme presented in [3], we are also interested in studying the impact of our solution on other GVHO schemes.

References

[1] N. C. Quental and P. A. S.Gonçalves, “A Policy for Group Vertical Handover Attempts,” in Proc. of the 13th International Conference on Networks (ICN), Nice, 2014, pp. 154–159.

[2] X. Cai and F. Liu, “Network Selection for Group Handover in Multi-access Networks,” in Proc. of the IEEE International Conference on Communications (ICC), Beijing, 2008, pp. 2164–2168.

[3] W. Lee and D. Cho, “Enhanced Group Handover Scheme in Multi-Access Networks,” IEEE Transactions on Vehicular Technology, vol. 60, no. 5, 2011, pp. 2389–2395.

[4] H. Jeong, J. Choi, H. Kang, and H. Youn, “An Efficient Group-Based Channel Scanning Scheme for Handover with IEEE 802.16e,” in Proc. of the 26th IEEE International Conference on Advanced Information Networking and Applications Workshops (WAINA), Fukuoka, 2012, pp. 639–644.

[5] M. Chowdhury, S. H. Chae, and Y. M. Jang, “Group Handover Management in Mobile Femtocellular Network Deployment,” in Proc. of the 4th International Conference on Ubiquitous and Future Networks (ICUFN), Phuket, 2012, pp. 162 – 165.

[6] A. Fu, S. Lan, B. Huang, Z. Zhu, and Y. Zhang, “A Novel Group-Based Handover Authentication Scheme with Privacy Preservation for Mobile WiMAX Networks,” IEEE Communications Letters, vol. 16, no. 11, 2012, pp. 1744–1747.

[7] S. Park, J. Yu, and J. Ihm, “A Performance Evaluation of Vertical Handoff Scheme between Mobile-WiMax and Cellular Networks,” in Proc. of 16th International Conference on Computer Communications and Networks (ICCCN), Honolulu, 2007, pp. 894 – 899.

[8] D. Kim, H. Shin, and D. Shin, “A network-based handover scheme for hierarchical mobile ipv6 over ieee 802.16 e,” in Proc. of the 10th International Conference on Advanced Communication Technology (ICACT), vol. 1. Gangwon-Do: IEEE, 2008, pp. 468–472.

[9] W. Shen and Q. Zeng, “Cost-function-based network selection strategy in integrated wireless and mobile networks,” IEEE Transactions on Vehicular Technology, vol. 57, no. 6, 2008, pp. 3778–3788.

[10] E. Stevens-Navarro, Y. Lin, and V. Wong, “An MDP-Based Vertical Handoff Decision Algorithm for Heterogeneous Wireless Networks,” IEEE Transactions on Vehicular Technology, vol. 57, no. 2, 2008, pp. 1243–1254.

[11] J. Yeh, J. Chen, and P. Agrawal, “Fast intra-network and cross-layer handover (finch) for wimax and mobile internet,” IEEE Transactions on Mobile Computing, vol. 8, no. 4, 2009, pp. 558–574.

[12] R. Y. Kim, I. Jung, X. Yang, and C.-C. Chou, “Advanced Handover Schemes in IMT-Advanced Systems,” IEEE Communications Magazine, vol. 48, no. 8, 2010, pp. 78–85.

[13] V. Gondi and N. Agoulmine, “Low latency handover and roaming using security context transfer for heterogeneous wireless and cellular networks,” in Proc. of the IEEE Asia-Pacific Services Computing Conference (APSCC). Hangzhou: IEEE, 2010, pp. 548–554.

[14] H. Choi, “An optimal handover decision for throughput enhancement,” IEEE Communications Letters, vol. 14, no. 9, 2010, pp. 851–853.

[15] S. Koh and M. Gohar, “Multicast Handover Agents for Fast Handover in Wireless Multicast Networks,” IEEE Communications Letters, vol. 14, no. 7, 2010, pp. 676–678.

[16] T.-S. Kim, R. Oh, S.-J. Lee, C.-H. Lim, S. Ryu, and C.-H. Cho, “Cell selection and trigger point decision for next generation heterogeneous wireless networking environment: Algorithm & evaluation,” in Proc. of the 5th International Conference on New Trends in Information Science and Service Science (NISS), Macao, 2011, pp. 247–254.

[17] I. Kim and Y. Kim, “Performance Evaluation and Improvement of TCP Throughput over PFMIPv6 with MIH,” in Proc. of the 12th IFIP/IEEE International Symposium on Integrated Network Management (IM), Dublin, 2011, pp. 997–1004.

[18] M. Zekri, B. Jouaber, and D. Zeghlache, “A Review on Mobility Management and Vertical Handover Solutions over Heterogeneous Wireless Networks,” Computer Communications, vol. 35, no. 17, 2012, pp. 2055–2068.

[19] K. Taniuchi, Y. Ohba, V. Fajardo, S. Das, M. Tauil, Y. Cheng, A. Dutta, D. Baker, M. Yajnik, and D. Famolari, “IEEE 802.21: Media Independent Handover: Features, Applicability, and Realization,” IEEE Communications Magazine, vol. 47, no. 1, 2009, pp. 112–120.

[20] S. Gundavelli, K. Leung, V. Devarapalli, K. Chowdhury, and B. Patil, “Proxy Mobile IPv6,” RFC 5213, aug 2008, [Accessed Nov. 11, 2014]. [Online]. Available: http://tools.ietf.org/html/rfc5213

[21] L. Shan, F. Liu, L. Wang, and Y. Ji, “Predictive Group Handover Scheme with Channel Borrowing for Mobile Relay Systems,” in Proc. of the International Wireless Communications and Mobile Computing Conference (IWCMC), Crete Island, 2008, pp. 153–158.

[22] S. Lei, T. Hui, and H. Zheng, “Group Vertical Handover in Heterogeneous Radio Access Networks,” in Proc. of the 72nd IEEE Vehicular Technology Conference Fall (VTC), Ottawa, 2010, pp. 1–5.

[23] D. Niyato and E. Hossain, “Dynamics of Network Selection in Heterogeneous Wireless Networks: an Evolutionary Game Approach,” IEEE Transactions on Vehicular Technology, vol. 58, no. 4, 2009, pp. 2008–2017.

[24] S. Lee, K. Sriram, K. Kim, Y. Kim, and N. Golmie, “Vertical Handoff Decision Algorithms for Providing Optimized Performance in Heterogeneous Wireless Networks,” IEEE Transactions on Vehicular Technology, vol. 58, no. 2, 2009, pp. 865–881.

[25] D. Johnson, C. Perkins, and J. Arkko, “Mobility Support in IPv6,” RFC 3775, june 2004, [Accessed Nov. 11, 2014]. [Online]. Available: http://tools.ietf.org/html/rfc3775

[26] E. R. Koodli, “Mobile IPv6 Fast Handovers,” RFC 5568, july 2009, [Accessed Nov. 11, 2014]. [Online]. Available: http://tools.ietf.org/html/rfc5568

[27] H. Yokota, K. Chowdhury, R. Koodli, B. Patil, and F. Xia, “Fast Handovers for Proxy Mobile IPv6,” RFC 5949, september 201, [Accessed Nov. 11, 2014]. [Online]. Available: http://tools.ietf.org/html/rfc5949

[28] K. P. C. Viho, A. Ksentini, and J.-M. Bonnin, “Resource Management in Mobile Heterogeneous Networks: State of the Art and Challenges,” INRIA, Tech. Rep., 2008.

[29] S. J. Bae, M. Y. Chung, and J. So, “Handover Triggering Mechanism Based on IEEE 802.21 in Heterogeneous Networks with LTE and WLAN,” in Proc. of the International Conference on Information Networking (ICOIN), Barcelona, 2011, pp. 399–403.

[30] M. Q. Khan and S. H. Andresen, “PoA Selection in 802.11 Networks Using Media Independent Information Server (MIIS),” in Proc. of the 26th International Conference on Advanced Information Networking and Applications Workshops (WAINA), Fukuoka, Japan, 2012, pp. 454–459.

[31] Y.-H. Liang, B.-J. Chang, and C.-T. Chen, “Media Independent Handover-based Competitive On-Line CAC for Seamless Mobile Wireless Networks,” Journal Wireless Personal Networks, 2011.

[32] R. Silva, P. Carvalho, P. Sousa, and P. Neves, “Heterogeneous Mobility in Next Generation Devices: An Android-Based Case Study,” Mobile Multimedia Communications, vol. 77, 2012, pp. 316–330.

[33] L. Ning, Z. Wang, Q. Guo, and K. Jiang, “Fuzzy Clustering based Group Vertical Handover Decision for Heterogeneous Wireless Networks,” in Proc. of the IEEE Wireless Communications and Networking Conference (WCNC), Changai, 2013, pp. 1231–1336.

[34] B. Kwak, N. Song, and L. Miller, “Performance Analysis of Exponential Backoff,” IEEE/ACM Transactions on Networking, vol. 13, no. 2, 2005, pp. 343–355.

[35] Transport Systems Generic Requirements (TSGR): Common Requirements. GR-499, Issue 2, Telecordia Report 2009.

[36] M. K. Karray, “Evaluation of the Blocking Probability and the Throughput in the Uplink of Wireless Cellular Networks,” in Proc. of the International Conference on Communications and Networking (ComNet), Tozeur, Tunisia, 2010, pp. 1–8.

[37] C. Chekuri, K. Ramanan, P. Whiting, and L. Zhang, “Blocking Probability Estimates in a Partitioned Sector TDMA System,” in Proc. of the 4th International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications (DIAL-M), Boston, USA, 2000, pp. 28–34.

[38] H. Khedher, F. Valois, and S. Tabbane, “Channel Holding Time Characterization in Real GSM Network,” in Proc. of the 14th International Symposium on Personal,Indoor and Mobile Radio Communication (PIMRC), Beijing, 2003, pp. 46–49.

[39] Y. Zhang and B.-H. Soong, “Channel Holding Time in Hierarchical Cellular Systems,” IEEE Communications Letters, vol. 8, no. 10, 2004, pp. 614–616.

[40] G. Haring, R. Marie, R. Puigjaner, and K. Trivedi, “Loss Formulas and Their Application to Optimization for Cellular Networks,” IEEE Transactions on Vehicular Technology, vol. 43, no. 3, 2001, pp. 664–673.

Application-Aware Bandwidth Scheduling for Data Center Networks

Andrew Lester[1] Yongning Tang[2] Tibor Gyires[2]
[1]*Cloud Networking Group. Cisco Systems, Inc. - San Jose, CA, USA*
[2]*School of Information Technology. Illinois State University. Normal, IL. USA*
aeleste@cisco.com, ytang@ilstu.edu, tbgyires@ilstu.edu

***Abstract*—Recent study showed that many network applications require multiple different network flows to complete their tasks. Provisioning bandwidth to network applications other than individual flows in data center networks is becoming increasingly important to achieve user satisfaction on their received network services. Modern data center networks commonly adopt multi-rooted tree topologies. Equal-Cost Multi-Path (ECMP) forwarding is often used to achieve high link utilization and improve network throughput. Meanwhile, max-min fairness is widely used to allocate network bandwidth fairly among individual network flows. Today's data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements. In this paper, we first propose a flow-based scheduling mechanism (called *FlowSch*) to provide a prioritized Max-Min fair multiple path forwarding to improve link utilization and improve application performance. Then, we demonstrate and discuss that *FlowSch* may not perform effectively when network applications commonly use multiple network flows to accomplish their tasks. Accordingly, we design an application-aware scheduling mechanism (called *AppSch*) to tackle this challenge. *AppSch* can optimally allocate available bandwidth to satisfy application requirements. Our performance evaluation results show that *FlowSch* can improve flow throughput 10-12% on average and increase overall link utilization especially when the total demanded bandwidth is close or even exceeds the bisectional bandwidth of a data center network. However, when most applications rely on multiple network flows, *AppSch* can improve link utilization more effectively and reduce the application completion time 36-58%.**

Keywords- application-aware; SDN; max-min fair; scheduling.

I. INTRODUCTION

Elastic cloud computing is becoming pervasive for many emerging applications, such as big data online analysis, virtual computing infrastructure, and various web applications. Various cloud applications commonly share the same network infrastructure [2] [4] [29] in a data center, and compete for the shared resource (e.g., bandwidth). Many of these emerging cloud applications are complex combinations of multiple services, and require predictable performance, high availability, and high intra-data center bandwidth. For example, Facebook "experiences 1000 times more traffic inside its data centers than it sends to and receives from outside users", and the internal traffic has increased much faster than Internet-facing bandwidth [40]. Meanwhile, many data center networks are oversubscribed, as high as 40 : 1 in some Facebook data centers [41], causing the intra-data center traffic to contend for core bandwidth. Hence, providing bandwidth guarantees to specific applications is highly desirable, in order to preserve their response-time predictability when they compete for bandwidth with other applications.

The challenge of achieving high resource utilization makes cloud service providers under constant pressure to guarantee quality of service and increase customer satisfaction.

A Data Center (DC) refers to any large, dedicated cluster of computers that is owned and operated by a single authority, built and employed for a diverse set of purposes. Large universities and private enterprises are increasingly consolidating their Information Technology (IT) services within on-site data centers containing a few hundred to a few thousand servers. On the other hand, large online service providers, such as Google, Microsoft, and Amazon, are rapidly building geographically diverse cloud data centers, often containing more than 10,000 servers, to offer a variety of cloud-based services such as web servers, storage, search, on-line gaming. These service providers also employ some of their data centers to run large-scale data-intensive tasks, such as indexing Web pages or analyzing large data-sets, often using variations of the MapReduce paradigm.

Many data center applications (e.g., scientific computing, web search, MapReduce) require substantial bandwidth. With the growth of bandwidth demands for running various user applications, data centers also continuously scale the capacity of the network fabric for new all-to-all communication patterns, which presents a particular challenge for traditional data forwarding (switching and routing) mechanisms. For example, MapReduce based applications, as a currently adopted default computing paradigm for big data, need to perform significant data shuffling to transport the output of its map phase before proceeding with its reduce phase. Recent study shows the principle bottleneck in large-scale clusters is often inter-node communication bandwidth. Traffic pattern study [27] showed that only a subset (25% or less) of the core links often experience high utilization.

The disruptive Software-Defined Networking (SDN) technology shifts today's networks that controlled by a set of vendor specific network primitives to a new network paradigm empowered by new programmatic abstraction. OpenFlow provides a protocol such that the logical centralized controller can exploit forwarding tables on SDN switches for programmatic multi-layer forwarding flexibility. One of the fundamental transformations that flow based forwarding presents is the inclusion of multi-layer header information to make forwarding match and action logic programmatically. Programmatic policy is vital to manage the enormous combinations of user requirements. For example, an SDN controller can flexibly define a network flow using a tuple as (incoming port, MAC Src, MAC Dst, Eth Type, VLAN ID, IP Src, IP Dst, Port Src, Port Dst, Action), or schedule specific flows onto desired network paths. With the new flexibility and capability on network traffic manipulation empowered by SDN, various new network architecture and control mechanisms have been proposed for data center networks to optimize their resource allocation.

Modern data center networks commonly adopt multi-rooted tree topologies [2] [4] [29]. ECMP is often used to achieve high link utilization and improve network throughput. Meanwhile, max-min fairness is widely used to allocate network bandwidth fairly among multiple applications. Many current data center schedulers, including Hadoops Fair Scheduler [37] and Capacity Scheduler [35], Seawall [34], and DRF [38], provide max-min fairness. The attractiveness of max-min fairness stems from its generality. However, today's data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements.

In this paper, we first propose a Flow-based Scheduling mechanism (called *FlowSch*) to provide a prioritized Max-Min fair multiple path forwarding to improve link utilization and flow-based throughput. *FlowSch* can optimally allocate current available bandwidth to satisfy user demands specified by per flow. When predefined user requirements are available, *FlowSch* can prioritize current demands and allocate available bandwidth accordingly. Then, we demonstrate and discuss that *FlowSch* may not perform effectively when network applications commonly use multiple network flows to accomplish their tasks. Accordingly, we design an Application-Aware Scheduling mechanism (called *AppSch*) to tackle this challenge. Our evaluation shows that *AppSch* can optimally allocate available bandwidth to satisfy application requirements.

The rest of the paper is organized as the following. Section II discusses the related research work. Section III describes *FlowSch*. Section IV formalizes the application-aware scheduling problem and presents our solution *AppSch*. Section V presents our simulation design and results, respectively. Finally, Section VI concludes the paper with future directions.

II. Related Work

Current large data center networks connect multiple Ethernet LANs using IP routers and run scalable routing algorithms over a number of IP routers. These layer 3 routing algorithms allow for shortest path and ECMP routing, which provide much more usable bandwidth than Ethernets spanning tree. However, the

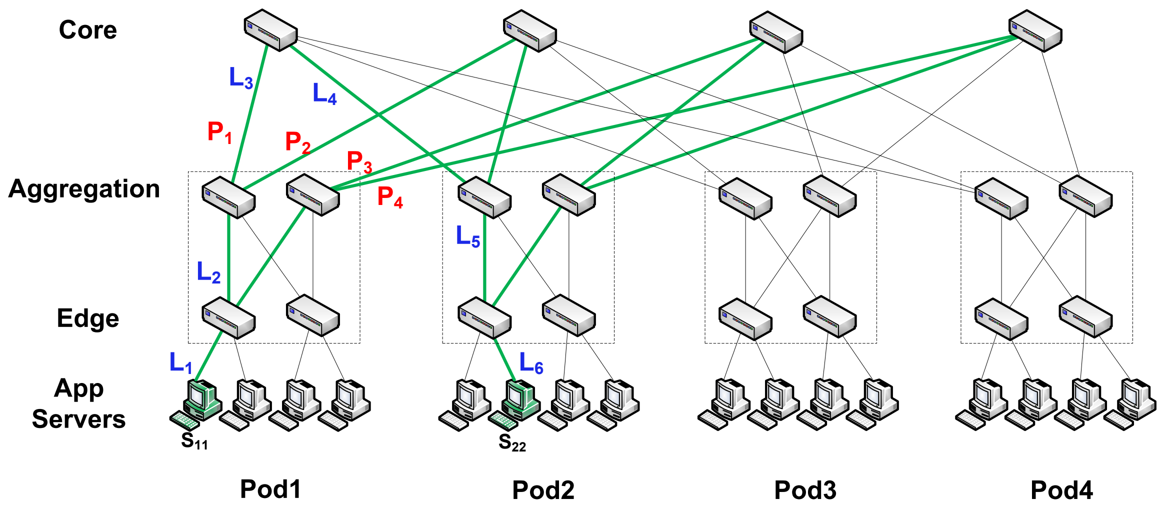


Figure 1. Fat tree topology.

mixed layer 2 and layer 3 solutions require significant manual configuration.

The trend in recent works to address these problems is to introduce special hardware and topologies. For example, PortLand [4] is implementable on Fat Tree topologies and requires ECMP hardware that is not available on every Ethernet switch. TRILL [5] introduces a new packet header format and thus requires new hardware and/or firmware features.

There have been many recent proposals for scale-out multi-path data center topologies, such as Clos networks [6] [8], direct networks like HyperX [9], Flattened Butterfly [11], DragonFly [12], etc., and even randomly connected topologies have been proposed in Jellyfish [16].

Many current proposals use ECMP-based techniques, which are inadequate to utilize all paths, or to dynamically load balance traffic. Routing proposals for these networks are limited to shortest path routing (or K-shortest path routing with Jellyfish) and end up under utilizing the network, more so in the presence of failures. While DAL routing [9] allows deroutes, it is limited to HyperX topologies. In contrast, Dahu [29] proposes a topology-independent, deployable solution for non-minimal routing that eliminates routing loops, routes around failures, and achieves high network utilization.

Hedera [17] and MicroTE [22] propose a centralized controller to schedule long lived flows on globally optimal paths. However, they operate on longer time scales and scaling them to large networks with many flows is challenging. Techniques like Hedera, which select a path for a flow based on current network conditions, suffer from a common problem: when network conditions change over time the selected path may no longer be the optimal one. While DevoFlow [24] improves the scalability through switch hardware changes, it does not support non-minimal routing or dynamic hashing. Dahu can co-exist with such techniques to better handle congestion at finer time scales.

MPTCP [19] proposes a host based approach for multi-path load balancing by splitting a flow into multiple sub flows and modulating how much data is sent over different subflows based on congestion. However, as a transport protocol, it does not have control over the network paths taken by subflows. Dahu [29] exposes the path diversity to MPTCP and enables MPTCP to efficiently utilize the non-shortest paths in a direct connect network. There have also been proposals that employ variants of switch-local per-packet traffic splitting [30].

Traffic engineering has been well studied in the context of wide area networks. TeXCP [31] and REPLEX [32] split flows on different paths based on load. However, their long control loops make them inapplicable in the data center context that requires faster response times to deal with short flows and dynamic traffic changes. PDQ [10] and pFabric [36] can support a scheduling policy like shortest flow first

(SFF), which minimizes flow completion times by assigning resources based on flow sizes. FLARE [11] exploits the inherent burstiness in TCP flows to schedule "flowlets" (bursts of packets) on different paths to reduce extensive packet reordering.

In our previous work [1], a flow-based bandwidth scheduling approach has been introduced. Several recent work [3] [7] [14] contributed to task-Aware Schedulers and network Abstractions. Orchestra [13] and CoFlow [7] argued for bringing task awareness in data centers. Orchestra focuses on how task awareness could provide benefits for MapReduce style workloads, and focuses on improvement in the average task completion time for batched workload. Baraat [3] makes the scheduling decisions in a decentralized fashion based on a revised FIFO mechanism. Baraat can also improve the tail completion time, for dynamic scenarios and multi-stage workloads. CoFlow [7] focuses on a new abstraction that can capture rich task semantics, which is orthogonal to Baraats focus on scheduling policy and the underlying mechanism. However, beyond the abstraction, CoFlow does not propose any new scheduling policy or mechanism to achieve task-awareness.

III. Flow-based Prioritized Max-Min Fair Bandwidth Scheduling

While ECMP is often used to achieve high link utilization, max-min fairness is widely used to allocate network bandwidth fairly among multiple applications. However, today's data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements. We propose Prioritized Max-Min Fair Multiple Path forwarding (*FlowSch*) to tackle this challenge. In the following, we first formalize the problem, and then present how *FlowSch* works.

A. Problem Formalization

Consider a data center network with K-ary fat-tree topology as shown in Fig.1, composed of a set of core switches S_c, a set of aggregation switches S_a, a set of edge switches S_e, and a set of hosts H. Each switch has k-port. There are k pods. Each pod contains $k/2$ aggregation switches and $k/2$ edge switches. In each pod, each k-port edge switch is directly connected to $k/2$ hosts and $k/2$ aggregation switches. The i^{th} port of each core switch $s_i \in S_c (i \in [1,(k/2)^2])$ is connected to pod i [4]. We assume all links (e.g., L_1 in Fig.1) have the same bandwidth for both uplink (e.g., L_1^u) and downlink (e.g., L_1^d) connections.

Recent study [27] showed that less than 25% of the core links have been highly utilized while packet losses and congestions may still often occur. In this paper, we only focus on inter-pod network traffic that requires bandwidth from core links. We denote all links between aggregation and core layers as a set L_{ac}, all links between edge and aggregation layers as a set L_{ea}, and all links between application server and edge layers as a set L_{se}. Generally, in a network with K-ary fat-tree topology , there are k paths between any two hosts from different pods.

```
Input: A list of tasks $\{T_i\}$; current link utilization $U(L_j)$
Output: Path assignment $PA$ with $PA_i$ for each task $T_i$

1:  Sort $\{T_i\}$ based on their priority levels $K_i$
2:  Start from the highest priority $W = m$ /*$m$ is the highest priority level*/
3:  for all $T_i! = \emptyset$ $(PL(T_i) = W)$ do
4:      /*The function $PL()$ returns the priority level of a given task*/
5:      Find all paths for each task $T_i$
6:      Assign a unit bandwidth (UB) to the least utilized path for each task /*we choose UB = 100Kbps*/
7:      $PA_i \leftarrow \{T_i, \{P_i\}\}$
8:      $PA \leftarrow PA \cup \{PA_i\}$
9:      if A path $P$ is saturated and $P \in APL(T_i)$ then
10:         $APL(T_i) \leftarrow APL(T_i) - P$
11:     end if
12:     if $APL(T_i) == \emptyset$ then
13:         Remove $T_i$
14:     end if
15:     if $(\{T_i\} == \emptyset)$ and $(W > 1)$ then
16:         $W = m - 1$
17:     end if
18: end for
19: return $PA$
```

Figure 2. Multi-Level progressive filling algorithm

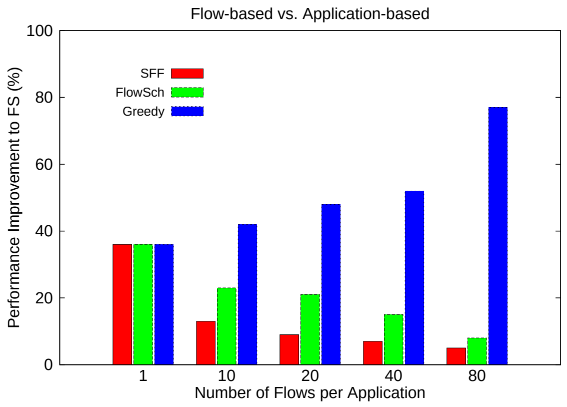


Figure 3. Completion time comparison

A network task T_i is specified by a source and destination hosts (e.g., S_{11} and S_{22}) and the expected traffic volume. We also consider each task with different priority level w_i. Here, $w_i \in [1, m]$ with the lowest and highest priority levels as 1 and m, respectively. A network scheduler modular (also simply referred to as scheduler) on an SDN controller needs to decide how to allocate available bandwidth to maximally satisfy the application requirements. We define a valid Network Path Assignment PA_i for a given task T_i is a set of paths and their corresponding allocated bandwidths connecting the source to the destination (e.g., a subset of $\{P_1, P_2, P_3, P_4\}$), in which each path consists of a list of directional links (e.g., $P_1 = \{L_1^u, L_2^u, L_3^u, L_4^d, L_5^d, L_6^d\}$) connecting the source to the destination hosts. Here, $L_1^u, L_6^d \in L_{se}$; $L_2^u, L_5^d \in L_{ea}$; $L_3^u, L_4^d \in L_{ac}$.

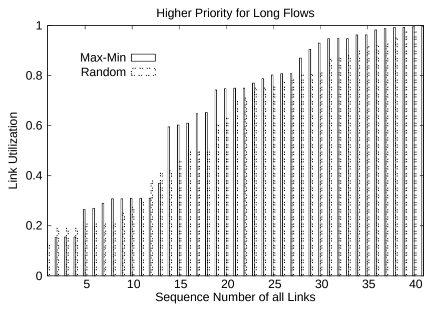
There is a variety of applications on a data center network, which have different service requirements regarding throughput, packet loss, and delay. For our analysis, we characterize the applications' requirements through their priority levels, which can be the output of some utility function. Priorities can offer a basis for providing application and business oriented service to users with diverse requirements. We consider a model where the weight associated with the different priority classes is user-definable and static. Users can freely define the priority of their traffic, but are charged accordingly by the network. We aim to study the bandwidth-sharing properties of this priority scheme. Given a set of network tasks $T = \{T_i\}$ $(i \geq 1)$ and their corresponding priority levels $K = \{K_i\}$, we consider a Network Path Assignment problem is to find a set of path assignment $PA = \{PA_i\}$ to satisfy the condition of Prioritized Max-Min Fairness.

Definition 1. Prioritized Max-Min Fairness A feasible path assignment PA^x is "prioritized max-min fair" if and only if an increase of any path bandwidth within the domain of feasible bandwidth allocations must be at the cost of a decrease of some already less allocated bandwidth from the tasks with the same or higher priority level. Formally, for any other feasible bandwidth allocation scheme PA^y, if $BW(PA^y_{T_i}) > BW(PA^x_{T_i})$, then it decreases the allocated bandwidth of some other path with the same or higher priority level. Here, $BW(PA^y_{T_i})$ is the total allocated bandwidth for the task T_i in the bandwidth allocation scheme PA^y.

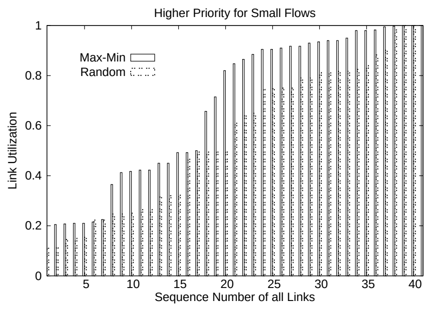
Definition 2. Saturated Path A path P_i is saturated if at least one bottleneck link L_j exists on the path P_i. A link is bottlenecked if the total assigned bandwidth on this link from the given tasks is more than or equal to the maximum bandwidth of the link. Formally, a bottleneck link is the one that $\sum_i BW_{T_i}(L_j) \geq BW_{max}(L_j)$.

B. The Algorithm of Multi-Level Progressive Filling

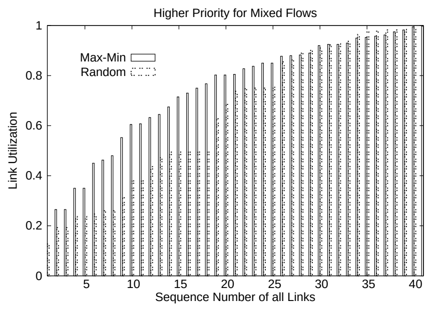
The network tasks can be dynamically and continuously generated, and submitted to the scheduler. In *FlowSch*, the scheduler can periodically query all network switches to collect current link utilizations. Once a new task list received, the scheduler will use a practical approach called "progressive filling" [33] provisioning available bandwidth that results in a prioritized max-min fair allocation following the priority order from the highest to the lowest priority level. The idea is shown in Fig. 2: The scheduler starts with all provisioned bandwidth equal to 0 and increases all bandwidths together at the same pace for the tasks with the same priority level until one or several saturated paths are found. The bandwidth for the corresponding tasks that use these paths are not increased any more and the scheduler continue increasing the bandwidth for other tasks on the same priority level. All the tasks that are stopped have a saturated path. The algorithm continues until it is not possible to increase the bandwidth for the tasks at certain priority level. Then, the algorithm moves to the next priority level and repeats the same bandwidth provisioning operations until all tasks are assigned



(a)

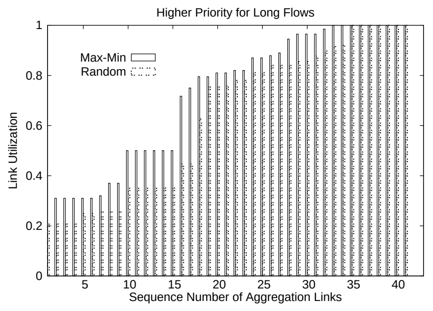


(b)

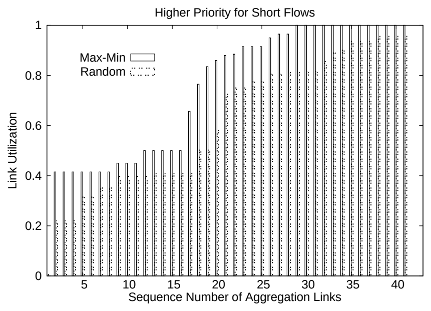


(c)

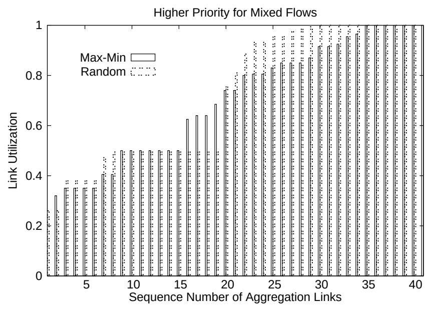
Figure 4. Path utilization with higher priority for (a) long flows (b)short flows (c)mixed flows



(a)



(b)



(c)

Figure 5. Aggregation link utilization with higher priority for (a) long flows (b)short flows (c)mixed flows

to some paths. The algorithm terminates because the total paths and tasks are finite. When the algorithm terminates all tasks have been served at some time and thus have a saturated path. By Definition 1 the allocation is max-min fair for the tasks at the same priority level.

$$\min \sum_{k=1}^{|A|} T_k \tag{1}$$

such that:

$$\sum_{j=1}^{|P|} x_{ij}^k = 1 \tag{2}$$

$$x_{ij}^k \in \{0,1\} \tag{3}$$

$$B_j \in \{0,1\} \tag{4}$$

IV. Application-based Scheduling

Recent study [38] showed that 70% of applications involve 30-100 flows, 2% involve more than 150 flows. In a multi-flow based application, the application flows may traverse different parts of the network and not all of them may be active at the same time. Only after all these related flows finish, the corresponding application finishes and the user gets a response. The distributed nature and scale of data center applications results in rich and complex work flows. Typically, these applications run on many servers that, in order to respond to a user request, process data and communicate across the internal network. Traditionally, allocation of network bandwidth has targeted per-flow fairness. Because latency is the primary goal for many data center applications, recent proposals [21] [36] indicate that per-flow fairness scheduling that optimizes flow-level metrics (e.g., minimizing flow completion time) may not necessarily improve user perceivable application performance. Typical data center application applications can have many flows, potentially of different sizes. Flow-based scheduling presents some inefficiency when applied to applications relying on multiple flows.

In the following, we first present the result of a simple simulation that demonstrates the different effect on application performance (i.e., completion time) improvement with three different scheduling algorithms, namely Shortest Flow First

(*SFF*), *FlowSch*, and an off-line greedy scheduling algorithm (*Greedy*). Then, we tackle the inefficiency of flow-based scheduling by introducing a new application-aware scheduling approach called *AppSch*.

A. Flow-based vs. Application-based Scheduling

FlowSch is designed as an application agnostic scheduler that targets per-flow fairness among applications with the same priority, which may not improve the performance of multi-flow based applications.

We validate this through a simple simulation that compares performance improvement in terms of application completion times with three different approaches, namely Shortest Flow First (*SFF*), *FlowSch*, and an off-line greedy scheduling algorithm (*Greedy*). *SFF* schedules the shorter flows of every task first, leaving longer flows to the end. This can hurt application performance by delaying completion of tasks. *FlowSch* considers max-min fair sharing among all flows that allocates resources with a lower bound. *Greedy* is an off-line greedy algorithm searching for the “best” assignments for all flows, which also shows the room for improvement.

In this simulation, we use a simple single-stage partition-aggregate work flow scenario [7] with 60 applications comprising flows uniformly chosen from the range $[5, 40]$ KB. Fig. 5 shows SFFs improvement over fair-sharing as a function of the number of flows in a application. If an application has just a single flow, SFF reduces the application completion time by almost 40%. However, as we increase the number of flows per application, the benefits reduce. The same observation also occurred with FlowSch, which however, outperforms SFF due to its max-min sharing. Comparing to the “best” scheduling offered by the off-line greedy scheduling algorithm, application agnostic flow-based scheduling does not perform well in terms of the improvement on application completion time, comparing to the performance of an off-line application-aware scheduler referred to as *Greedy* in Fig. 5.

B. Application Requirements Abstraction

Although many data-intensive applications are network-bound [13] [17], network scheduling remains agnostic to application specific network requirements. In recent work, Coflow [7] argues for tasks (or Coflows) as a first-order abstraction for the network data plane to compensate the mismatch that often affects application-level performance, even when network-oriented metrics like flow completion time (FCT) or fairness improve. The recently proposed coflow abstraction [7] represents such collections of parallel flows to convey application-specific network requirements, for example, minimizing completion time or meeting a deadline to the network and enables application-aware network scheduling.

Allowing applications to expose their semantics to the network could significantly help the network optimize its resource allocation for application-level metrics. For example, allocating network bandwidth to applications in a FIFO fashion, such that they are scheduled over the network one at a time, can improve the average application completion time as compared to per-flow fair sharing (e.g., TCP).

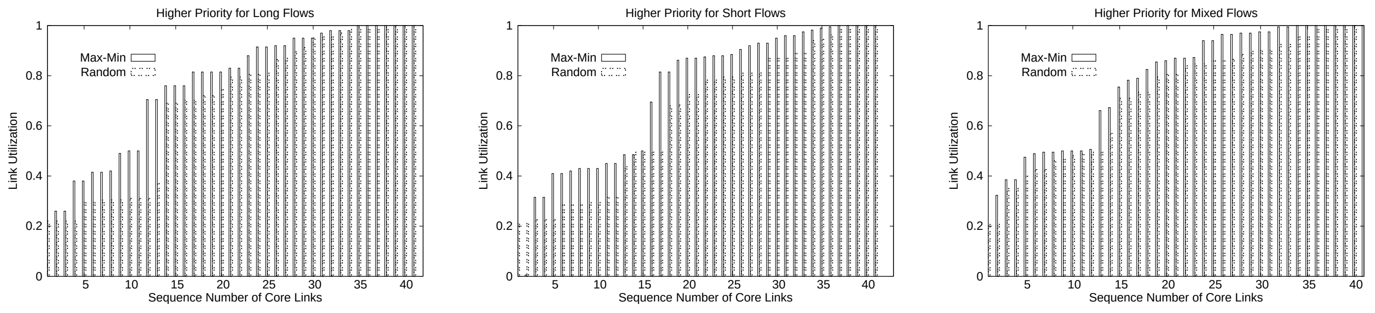
In this paper, we characterize two features of application tasks in todays data centers: 1) the task size, and 2) the number of flows per task. Both information are critical when considering application-aware scheduling for the network; the first influences the scheduling policy, while the latter governs when application-aware scheduling outperforms flow-based scheduling.

In the following, we formalize the application-aware scheduling as a variant bin packing problem, and presents a heuristic algorithm to tackle this NP-hard problem.

C. Bin Packing with Varying Capacities

We assume that the amount of data each flow in an application needs to transfer is known before it starts [13] [23] [36]. Analysis of production application traces [14] shows wide variations in application flow characteristics in terms of total size, the number of parallel flows, and the size of individual flows. But commonly these applications can be modeled as an ordered flow request list.

Let A be a list of applications A_i to be scheduled. Each application A_i has a list of flow volumes $V_i = \{v_{i1}, \cdots, v_{ik}\}$. Let $P = \{P_1, \cdots, P_m\}$ be the set of available network paths and let B_i be the current available bandwidth for path P_i. Without any loss of generality, we assume that the bandwidths associated with the network paths are integers.



(a) (b) (c)

Figure 6. Core link utilization with higher priority for (a) long flows (b)short flows (c)mixed flows

We define the path-selection variables $S = (s_1, \cdots, s_m)$, where $s_j = 1$ if path p_j is selected and $s_j = 0$, otherwise; and the flow-to-path assignment variables x_{ij}^k, where $x_{ij}^k = 1$ if flow f_{ik} from application A_k is scheduled on path P_j and $x_{ij}^k = 0$, otherwise. We want to schedule all application related flows to optimally utilize all available bandwidth, so as to minimize the total application completion time (T). For application A_k, the corresponding application completion time $T_k = \sum_{j=1}^{m} v_{ij}/(x_{ij}^k * B_j)$.

The objective function (1) minimizes the total amount of application completion time. Constraint (2) ensure that each flow request is assigned exactly to one network path; and constraints (3) and (4) enforce the integrality requirements for all decision variables.

The scheduling policy determines the order in which applications are scheduled across the network paths. Determining an ordering that minimizes application completion time can be easily reduced to a bin packing problem with varying bin sizes, which is NP-hard. Some previous similar work like flow-shop scheduling [14] [25], is considered as one of the hardest NP-hard problems, with exact solutions not known for even small instances of the problem [15]. Thus, we need to consider heuristic scheduling policies. The heuristic policy should help reduce both the average as well as tail application completion time. Guided by flow-based policies that schedule flows one at a time [17], we consider serving applications one at a time. This can help finish applications faster by reducing the amount of contention in the network. Consequently, we define application packing as the set of policies where an entire application is scheduled before moving to the next.

Specifically, for such a bin packing NP-hard problem [18], we design a heuristic algorithm called AppSch that adapts the well-known Best First Decreasing loading heuristic [15] and extends a number of fundamental concepts [15] [18] in the bin covering and knapsack methodology. AppSch first sorts all application flow requests according to the non-increasing order of their data sizes, and then sequentially assigns them into the path with the maximum available bandwidth. For each flow request, AppSch first attempts to assign it into the "best" already-selected path to increase the path utilization. If the flow request cannot be assigned to an already-selected path, a new path is selected and the flow request is assigned to it. One challenge in this problem different from classic bin packing problem where all bins are homogeneous, is how to choose a new path when required. Inspired by the item-selection rule for knapsack problems, we select paths according to the non-increasing order of the ratios of their data sizes and available path bandwidths, and in the non-decreasing order of their data sizes when the data sizes are equal.

V. Evaluation

In this section, we present our evaluation metrics, methodology and evaluation results.

A. Data Center Network Traffic Pattern

Several recent studies [26] [27] [28] have been conducted in various data center networks to understand network traffic patterns. The studied data center networks include university campus, private enterprise data centers, and cloud data centers running Web services, customer-facing applications, and intensive Map-Reduce jobs. The studies have

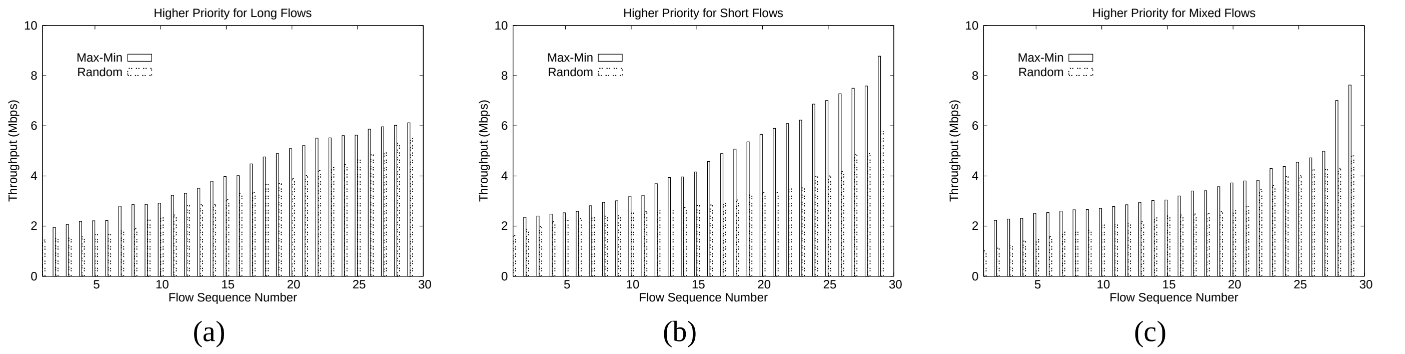


Figure 7. Throughput with higher priority for (a) long flows (b)short flows (c)mixed flows

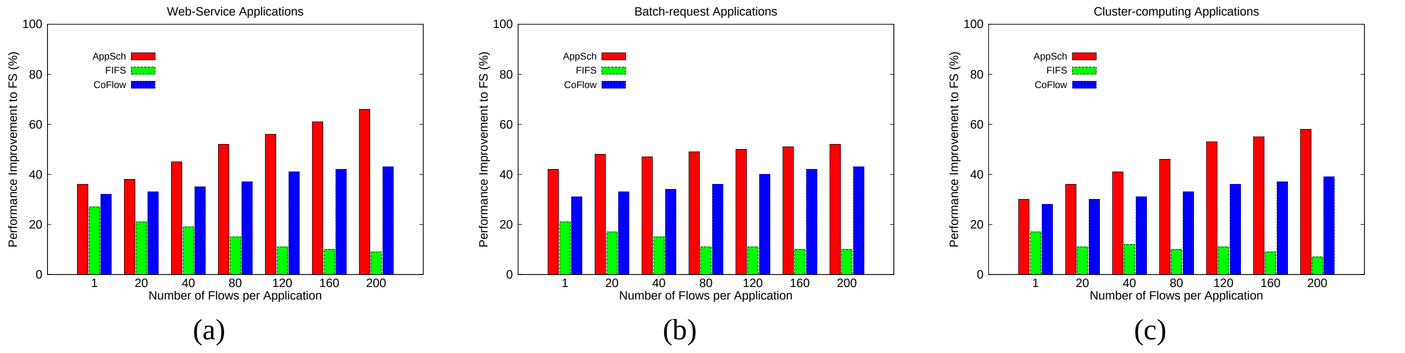


Figure 8. Application completion time for (a) Web Services (b) Batch Requests (c) Cluster Computing

shown some interesting facts: (1) The majority of the traffic in data center networks is TCP flows. (2) Most of the server generated traffic in the cloud data centers stays within a rack, while the opposite is true for campus data centers. (3) At the edge and aggregation layers, link utilizations are fairly low and show little variation. In contrast, link utilizations at the core network are high with significant variations over the course of a day. (4) In some data centers, a small but significant fraction of core links appear to be persistently congested, but there is enough spare capacity in the core to alleviate congestion. (5) Losses on the links that are lightly utilized on the average can be attributed to the bursty nature of the underlying applications run within the data centers.

B. Methodology and Metrics

In our experiments, we simulate a data center with a fat-tree topology. We implemented *FlowSch* based on RipL [39], a Python library that simplifies the creation of data center code, such as OpenFlow network controllers, simulations, or Mininet topologies. We compared *FlowSch* scheduler with a commonly used randomization based scheduling method.

In our evaluation, we use three different priority policies for a mixture of traffic patterns: (1) high priority for long TCP flows with the total data size between $1MB$ and $100MB$; (2) high priority for short TCP flows with the total data size between $10KB$ and $1MB$; (3) high priority for random selected flows including both short and long ones referred to as mixed TCP flows.

We focus on two performance metrics: (1) Link Utilization that demonstrates how effectively the scheduler utilizes the network bandwidth. Intuitively, when there are high bandwidth demands from user applications, the overall link and path utilizations should be kept in high. (2) Network throughput that shows how efficiently the network serves different applications.

For evaluating the performance of application-aware scheduler $AppSch$, we setup different application scenarios to mimics (1) web-service: a typical web-service scenario with one pod dedicated to the front-end nodes, while the other pods are used as caching back-end. For the experiment, we consider an online scenario where each user independently receives requests based on a Poisson arrival process. Each request (or application) corresponds to a multi-get that involves fetching data from randomly chosen back-end servers; (2) batched requests: to evaluate the impact of varying the number of concurrent applications in the system. For this experiment, one pod acts as a client while the other pods in the network act as storage servers. For the request, the client retrieves $100 - 800$ KB chunks from each of the servers. The request finishes when data is received from all servers; and (3) cluster computing: our workload is based on a Hive/MapReduce trace collected from a tier-1 ISP IDS system. We consider jobs with non-zero shuffle and divide them into bins based on the fraction of their durations spent in shuffle.

C. Link Utilization

We created 16 test scenarios to evaluate *FlowSch* with different inter-pod traffic patterns. We ran 5 tests for each scenarios. In all test scenarios, the test traffic traversed all edge, aggregation, and core links. The results of multiple test runs from the same test scenario present similar results. In the following, we only report the result of one test run for each test scenario that created traffic between two pods in both directions. Under the same three different priority policies, Fig.4(a)$\sim$(c) shows the overall path utilization; Fig.5(a)$\sim$(c) shows the aggregation link utilizations; and Fig.6(a)$\sim$(c) shows the core link utilizations. Comparing to the randomization based scheduler, our algorithm 2 achieves high utilization on path level, aggregation and core link levels by: (1) dynamically observing all link utilization status, and (2) progressively filling the jobs of the same priority with the available bandwidth with the max-min fairness. The average gain on utilization is approximately improved from 59% to 66%. Note that with the increase of link utilization, idle bandwidth can be effectively utilized by demanding network applications, which can correspondingly improve their performance by reducing their network latencies.

D. Network Throughput

Once the overall utilization can be increased, we expect that the overall application throughput should also be improved. The experiment results presented some interesting results as shown in Fig.7(a)$\sim$(c). When we emulate more realistic application scenarios, where short and long TCP flows are randomly mixed together, our *FlowSch* scheduler obviously outperforms the performance of the random scheduler with about 10-12% improvement. In the scenario of the different policies favoring either short or long flows, our scheduler adopts max-min fairness, and thus, the average throughput has been improved from $2.52Mbps$ in the random scheduler and to $3.46Mbps$ in the max-min scheduler.

E. Application Completion Time

The minimum completion time of an application A_i can be attained as long as all flows in the application finish at time T_i. We choose three applications (1) Web service, (2) Batch request, and (3) Cluster computing application that represent typical application communication patterns in today's data center networks to validate $AppSch$ and compare with $FIFS$ and $CoFlow$ [7]. From the experiment results as shown in Fig. 8, AppSch performs steadily with evident application performance (completion time) improvement (36%-58%) for all different types of applications.

VI. Conclusion

The role of the data center network is becoming ever more crucial today, which is evolving into the integrated platform for next-generation data centers. Because it is pervasive and scalable, the data center network is developing into a foundation across which information, application services and all data center resources, including servers, storage are shared, provisioned, and accessed. Modern data center networks commonly adopt multi-rooted tree topologies. ECMP is often used to achieve high link utilization and improve network throughput. Meanwhile, max-min fairness is widely used to allocate network bandwidth fairly among multiple applications. However, today's data centers usually host diverse applications, which have various priorities (e.g., mission critical applications) and service level agreements (e.g., high throughput). It is unclear how to adopt ECMP forwarding and max-min fairness in the presence of such requirements. We propose Prioritized Max-Min Fair Multiple Path forwarding (*FlowSch*) to tackle this challenge. *FlowSch* can prioritize current demands and allocate available bandwidth accordingly.

Our performance evaluation results show that *FlowSch* can improve application throughput 10-12% on average and increase overall link utilization especially when the total demanded bandwidth close or even exceed the bisectional bandwidth of a data center network.

References

[1] A. Lester, Y. Tang, T. Gyires. “Prioritized Adaptive Max-Min Fair Residual Bandwidth Allocation for Software-Defined Data Center Networks,” In the Thirteenth International Conference on Networks (ICN), 2014.

[2] M. Al-Fares, A. Loukissas, and A. Vahdat, A Scalable, “Commodity Data Center Network Architecture,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2008

[3] F. R. Dogar, T. Karagiannis, H. Ballani, and A. Rowstron, “Decentralized Task-Aware Scheduling for Data Center Networks,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2014

[4] R. N. Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat, “PortLand: A scalable fault-tolerant layer 2 data center network fabric,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2009

[5] R. Perlman, “Rbridges: Transparent routing,” In IEEE Conference on Computer Communications (INFOCOM), 2004.

[6] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, “VL2: A Scalable And Flexible Data Center Network,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2009.

[7] M. Chowdhury and I. Stoica, “Coflow: A networking abstraction for cluster applications,” In ACM Hot Topics in Networks (HotNets) workshops, 2012.

[8] V. Liu, D. Halperin, A. Krishnamurthy, and T. Anderson, “F10: A Fault-Tolerant Engineered Network,” In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2013

[9] J. H. Ahn, N. Binkert, A. Davis, M. McLaren, and R. S. Schreiber, “HyperX: Topology, Routing, and Packaging of Efficient Large-Scale Networks,” In ACM Conference on High Performance Computing Networking, Storage and Analysis, 2009.

[10] C. Hong, M. Caesar, and P. Godfrey, “Finishing flows quickly with preemptive scheduling,” ACM SIGCOMM Computer Communication Review (CCR), 2012.

[11] J. Kim, W. J. Dally, and D. Abts, “Flattened butterfly: A Cost-efficient Topology for High-radix networks,” In ACM International Symposium on Computer Architecture (ISCA), 2007.

[12] J. Kim, W. J. Dally, S. Scott, and D. Abts, “Technology-Driven, Highly-Scalable Dragonfly Topology,” In ACM International Symposium on Computer Architecture (ISCA), 2008

[13] M. Chowdhury, M. Zaharia, J. Ma, M. I. Jordan, and I. Stoica, “Managing data transfers in computer clusters with Orchestra,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2011.

[14] M. Chowdhury, Y Zhong, and I. Stoica, “Efficient Coflow Scheduling with Varys,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2014.

[15] W T Rhee and M Talagrand, “Optimal bin covering with items of random size,” SIAM Journal on Computing. 1989.

[16] A. Singla, C.-Y. Hong, L. Popa, and P. B. Godfrey, “Jellyfish: Networking Data Centers Randomly,” In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2012.

[17] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, “Hedera: Dynamic Flow Scheduling for Data Center Networks,” In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2010.

[18] A. S. Fukunaga and R. E. Korf, “Bin Completion Algorithms for Multicontainer Packing, Knapsack, and Covering Problems,” Journal of Artificial Intelligence Research 28 (2007) pp. 393 $\sim$ 429

[19] M. Alizadeh, A. Greenberg, D. A. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, “Data center TCP (DCTCP),” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2010.

[20] T. Benson, A. Akella, and D. A. Maltz, “Network Traffic Characteristics of Data Centers in the Wild,” In ACM Internet Measurement Conference (IMC), 2010.

[21] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, “Better never than late: Meeting deadlines in datacenter networks,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2011.

[22] T. Benson, A. Anand, A. Akella, and M. Zhang, “MicroTE: Fine Grained Traffic Engineering for Data Centers,” In ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2011.

[23] C.-Y. Hong, M. Caesar, and P. B. Godfrey, “Finishing flows quickly with preemptive scheduling,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2012.

[24] A. R. Curtis, J. C. Mogul, J. Tourrilhes, P. Yalagandula,

P. Sharma, and S. Banerjee, “DevoFlow: Scaling Flow Management for High-Performance Networks,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2011.

[25] OpenFlow Switch Specification (Version 1.1), www.openflow.org/documents/openflow-spec-v1.1.0.pdf, (retrieved: Nov. 2014)

[26] S. Kandula, S. Sengupta, A. Greenberg, P. Patel, and R. Chaiken, “The Nature of Data Center Traffic: Measurements and Analysis,” In ACM Internet Measurement Conference (IMC), 2009.

[27] T. Benson, A. Akella, and D. A. Maltz, “Network Traffic Characteristics of Data Centers in the Wild,” In ACM Internet Measurement Conference (IMC), 2010.

[28] G. Wang, D. G. Andersen, M. Kaminsky, M. Kozuch, T. S. E. Ng, K. Papagiannaki, M. Glick, and L. Mummert, “Your data center is a router: The case for reconfigurable optical circuit switched paths,” In ACM Hot Topics in Networks (HotNets) workshops, 2009.

[29] S. Radhakrishnan, M. Tewari, R. Kapoor, G. Porter, and A. Vahdat, “Dahu: Commodity Switches for Direct Connect Data Center Networks,” In Proceedings of the 9th ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS13), October 2013

[30] D. Zats, T. Das, P. Mohan, D. Borthakur, and R. Katz, “DeTail: Reducing the Flow Completion Time Tail in Datacenter Networks,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2012.

[31] S. Kandula, D. Katabi, B. Davie, and A. Charny, “Walking the Tightrope: Responsive Yet Stable Traffic Engineering,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2005.

[32] S. Fischer, N. Kammenhuber, and A. Feldmann, “REPLEX: Dynamic Traffic Engineering Based on Wardrop Routing Policies,” In ACM Conference on emerging Networking EXperiments and Technologies (CoNEXT), 2006.

[33] A. Ghodsi, M. Zaharia, S. Shenker and I. Stoica, “Choosy: Max-Min Fair Sharing for Datacenter Jobs with Constraints,” In European Conference on Computer Systems (EuroSys), 2013.

[34] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, I. Stoica, and S. Shenker, “Dominant resource fairness: Fair allocation of multiple resource types,” In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2011

[35] Hadoop Capacity Scheduler, hadoop.apache.org/docs/r1.2.1/capacity_scheduler.html, (retrieved: Nov. 2014)

[36] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, “pfabric: Minimal near-optimal datacenter transport,” In ACM conference of the Special Interest Group on Data Communication (SIGCOMM), 2013.

[37] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, and I. Stoica, “Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling,” In European Conference on Computer Systems (EuroSys), 2010.

[38] A. Shieh, S. Kandula, A. Greenberg, C. Kim, and B. Saha, “Sharing the data center network,” In USENIX Symposium on Networked Systems Design and Implementation (NSDI), 2011.

[39] M. Casado, D. Erickson, I. A. Ganichev, R. Griffith, B. Heller, N. Mckeown, D. Moon, T. Koponen, S. Shenker, and K. Zarifis, “Ripcord: A modular platform for data center networking,” UC, Berkeley, Technical Report UCB/EECS-2010-93

[40] “Facebook Future-Proofs Data Center With Revamped Network,” http://tinyurl.com/6v5pswv, (retrieved: Nov. 2014)

[41] N. Farrington and A. Andreyev, “Facebook’s Data Center Network Architecture,” IEEE Optical Interconnects, 2013.

Overview and Future of Switching Solutions for Industrial Ethernet

György Kálmán, Dalimir Orfanus, and Rahil Hussain
ABB Corporate Research Norway
{gyorgy.kalman, dalimir.orfanus, rahil.hussain}@no.abb.com

***Abstract*—Industrial Ethernet is the preferred network technology in industrial green field deployments. Although the performance of these networks is superior compared to legacy fieldbuses, the complexity of possible installations is an issue in the field. Both selection and placement of active devices and efficiently running network services are causing problems in planning and also during the life of the network. This paper is giving an overview on implementation possibilities of switched network in industrial applications with respect to performance, features, logical architecture, and flexibility. It also shows the importance of time synchronization and presents current standardization work with focus on impact on industrial applications. An outlook for expected future possibilities is shown, including the use of Software Defined Networking (SDN).**

Keywords*—industrial Ethernet, switch, embedded, discrete, soft switch, forwarding, performance, QoS, SDN, 1588, 802.1AS

I. Introduction

This paper is an extended version of our paper at ICN 2014, An Overview of Switching Solutions for Wired Industrial Ethernet [1]. Compared to the original, it contains an extended state-of-the-art analysis, a section on time synchronization, which is one of the most important features for industrial applications, extended experiments and an outlook with an introduction on how *Software Defined Networking* (SDN) could be used in an industrial setting.

Ethernet is already the dominating technology at the control and higher levels of an automation network and is expected, along with wireless Ethernet, to be the primary choice in field installations.

Because of resource constraints and *Quality of Service* (QoS) requirements, most of the automation networks are implemented as *Local Area Networks* (LANs) (Figure 1). The motivation is twofold: first, this approach leads to a simpler network configuration as there are no routing tables and there are plenty of resources to utilize without the need to coordinate with external nodes or other actors. Second, as the operation is closer to the physical layer, the QoS parameters can be held within more strict boundaries.

The paper is structured as follows: the second section provides background overview on industrial Ethernet, then current top design priorities are presented in the third: topology and fourth: time synchronization. In the fifth section, the possible switch architectures are presented including performance requirements of different applications.

Then the possible architectural solutions are explained, with discrete, embedded and soft switches as main categories. Performance comparison is given based on our testbed measurements focusing on latency and jitter. A conclusion on possible fields of use for the discrete, embedded and soft solutions is given. As a possible evolution, an outlook on SDN for future industrial Ethernet planning and extensions towards using Layer 3 networks and wireless solutions is shown.

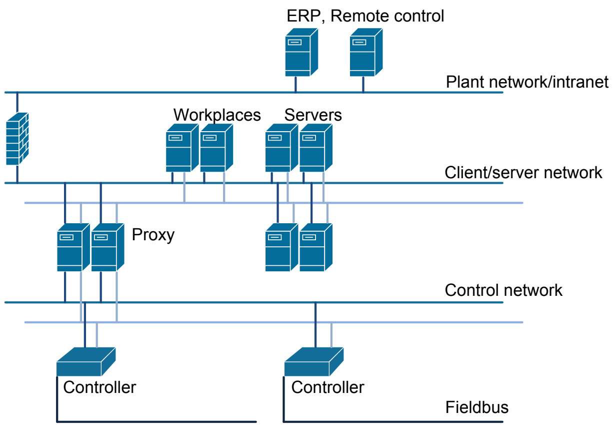


Fig. 1. An example of automation network architecture

II. Industrial Ethernet background

Industrial Ethernet enables the use of standard Ethernet devices and the IP protocol suite in automation networks [2]. By implementing industrial network based on Ethernet, vendors can create infrastructures that provide improved bandwidth, resiliency, and network security compared to fieldbus solutions (Figure 1). As an additional value, the use of already established standards lowers the risk associated with technology development.

A number of issues emerge from the fact that the Ethernet networks are replacing the fieldbuses. A heritage of the fieldbus past including design processes is the dominant cause of bus-like topologies (Figure 2) resulting in suboptimal operation of Ethernet [3]–[5].

The most challenging topology types are long chains of switches, which are often closed to form rings. While *Rapid Spanning Tree Protocol* (RSTP) was designed with loop-avoidance in mind, it is currently widely used redundancy protocol in the industry. In a ring structure, RSTP will disable one link and render the active network topology into a special tree, a line of switches.

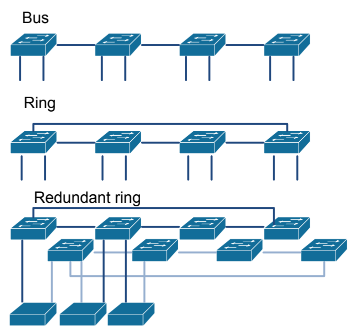


Fig. 2. Industrial Ethernet topologies

Although a line is a valid Ethernet topology and the technology will work, the industrial network will suffer from scalability issues due to much smaller end-node count than it could be expected from office experience [6], [7]. One of the reasons is that the most industrial Ethernet protocols are using standard Ethernet as bearer, with some exceptions such as PROFINET IRT or EtherCAT, so the network operation is in practice not different from the office counterparts. However, it runs on much less optimal infrastructure with more QoS sensitive applications. The other problem is raising in the requirements of precise timing. The need for synchronous operation is apparent, e.g., in case of synchrophasor operations, where the precise sampling of the waveform requires today a GPS time source. To enhance resiliency and to lower costs, time synchronization protocols are expected to provide a stable solution allowing the operation of such services even if the network is large and without the need to place an external precise time source at each critical network point.

The very long and sparse spanning tree is having a low branching factor (the average number of child links at each switch) and can lead to excess latency and jitter [8]–[11]. As compared to office counterparts, more manually configured nature of the industrial networks is creating another drawback: in a typical network the root bridge is configured manually and this node is typically not the one in the middle of the longest chain to break it into two halves. Thus, in calculating expected QoS parameters, the worst case having one single long line of nodes is used.

Still, in the most of the industrial operations, processes and automation tasks have tolerances several magnitudes higher than the total jitter of a reasonably built industrial Ethernet network could have. One of the exceptions is motion control where deadlines can be close to the jitter and/or delay of a large network, thus giving certain planning constraints. This particular field of automation shares requirements with applications in very different fields, e.g., finance and telecommunication.

In the current applications, however, automation tasks are focused on LAN operations, thus mostly the time-critical planning stops at the LAN-WAN interface. In telecom uses WAN environments are also included, which results in, e.g., different focus in standardization.

This convergence can clearly be seen from, e.g., the standardization efforts, where in addition to traditional networking and industrial companies, telcos and silicon vendors are taking a considerable effort.

III. Topology

In office environments, high port count switches are used to implement a high branching factor network, thus the issues associated with cascaded switches are less important [12], [13]. Also, in a typical setup, an office LAN is much earlier divided into subnetworks using firewalls and/or routers than reaching a deep spanning tree.

As an indirect result of the low branching factor and the pressure for lower costs and relatively high price of managed industrial Ethernet switches, the industrial installations are experiencing pressure from both sides: on the one, the required performance level is relatively high, especially for processing delays and jitter. On the other hand, the price of such devices on the market is relatively high in a project, because of the low port count and low general utilization of the resources. The recent trend is to include or integrate low port count switches directly into devices, i.e., devices can be interconnected into a cascade or a low branching factor tree without the need of other external devices [14]. Such topology has also practical impact on physical installation of cables: generally it is more far more expensive to lead a bundle of cables across the whole factory rather than a single cable that connects with the most or even all devices.

Moving towards switch as an integrated feature might lead to scalability problems because of increased latency and jitter on long sparse trees. This scalability issue is more apparent when integrated modules use low port count (2-3 ports, only providing enough ports for daisy-chaining). However, higher port count modules can be as flexible as the current situation with discrete devices in fieldbuses. The only limitation for the scalability of networks [15]–[17] can be when the daisy-chaining solution with few ports is combined with time-critical automation tasks.

Integrated switches are expected to lower the cost of building the network with lowering the amount of standalone devices and with better integration to the automation devices. The expected feature set is similar or equivalent to discrete units. In following sections we provide an overview of the typical embedded switch architectures and how they could be fitted into the industrial landscape.

IV. Time synchronization

One of the most important features in the current industrial deployments is the possibility and capability of precise time synchronization throughout the network.

The convergence between the industrial, telecom and finance requirements for a synchronous operation lead to that two standardization groups in IEEE are working on extending the current synchronization solutions towards additional networks and to widen their application areas.

Synchronizing time was always an important question over network applications, time references, timestamps are widely used in communication both for management, message exchange, logging and security applications. However, the granularity and the level of synchronization very much depends on the application. For less demanding tasks, e.g., process automation, a protocol with a fraction of a second precision would be sufficient. On the other end, (fine) motion control applications or power electronics would require sub microsecond level synchronization [18] to operate in a coordinated manner (Figure 3). Also in the incident management is important to be able to put the chain of events into the correct order.

Today the standard requirement for switched Ethernet is to support IEEE 1588v2 as the time synchronization protocol [19].

Another motivation for high precision time synchronization is also to filter out the non-deterministic processing delay in the active network components (switches, routers etc.). For this reason, IEEE 1588v2 supports Layer 2 time stamping of frames and different layers of procedures to enhance the precision.

The first place is in the network itself. Relay nodes in the network (switches, routers etc.) are introducing variable frame delay because they queue frames. This delay depends on the amount of traffic in the transit at a particular relay point. In IEEE 1588 the delay at a transit point can be calculated form timestamps put on ingress and egress frames.

The second level is the node level, which can be either an end node or a relay node. Timestamps are created in a way that the end node can calculate the correct time within a precision limit. In intermediate nodes, both at an ingress and egress ports the frame timestamps are updated to correct the transit time with the residence time in the node in order to keep the precision. The reason why IEEE 1588 and also other high precision synchronization protocols cannot rely solely on filtering solutions in upper layers is, that the measurement of time (here arrival and departure times) should be done as close to the physical medium as possible. Closeness to the physical world lowers uncertainties and allows for a more accurate measurement. Explicitly, IEEE 1588, if enabled, uses a hardware time stamper connected directly to the Ethernet card between the PHY and the MAC, to allow accurate measurement in the required precision range.

While this is a very efficient and good placement for a high precision stamping unit [20], the direct connection to wired Ethernet was also found to be a limitation, so in the future versions, it is expected that a hardware abstraction layer will be introduced to open for more networking technologies.

A. IEEE 1588v3

The work on the new version of the current standard Precision Time Protocol (PTP) is ongoing. The standard is expected to undergo a major overhaul compared to the v2 released in 2008 [22].

The new standard is expected to follow the example of the architecture of IEEE 802.1AS, which was originally developed for Audio/Video applications but currently evolving into a much more generic protocol under the IEEE Time Sensitive Networks group. To forecast the evolution of industrial Ethernet, it is important to get an view on how IEEE 1588v3 is expected to evolve compared to v2 [23].

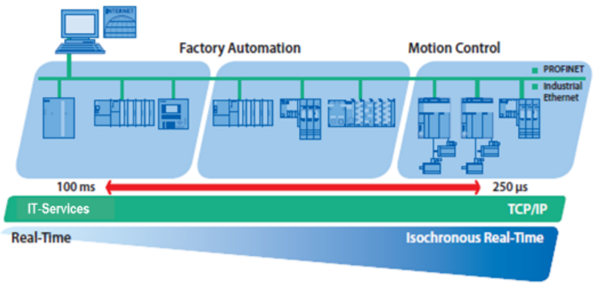


Fig. 3. Timing requirements for industrial Ethernet [21]

- *Common Management Information Base (MIB)*: The current 1588v2 landscape is somewhat disrupted as several parallel profiles exist, which are not fully compatible with each other. An example is the IEEE C37.238, the power profile, which uses a different MIB compared to other profiles, thus management of the complete time domain is more challenging.
- *A layered protocol structure*: The current PTP version uses functions which are stretching across network layers and thus limiting generic implementations of the protocol, where, e.g., the clock state machine can be reused even if the physical layer is exchanged.
- *High availability synchronization*: the protocol is planned to offer a solution for having multiple clocks in the time domain, where they can take over the master's role within specified limits.
- *Security*: IEEE 1588v3 is expected to provide a security mechanism to protect the time quality by using cryptographic functions for authenticity and integrity checking.
- *Wider scope*: The new PTP version is expected to support a wider range of networking technologies, including Wifi networks and wireless sensor networks. Support for time synchronization over IP is expected.

The most important change is to apply a proper protocol architecture where the media-dependent parts are connecting to the upper layers of the protocol through a standard interface. This enables 1588v3 to run on different networks.

Another difference w.r.t. 802.1AS is that 1588 does not require time-aware infrastructure. It can apply sophisticated filtering to reach good precision, although it would be less precise than with direct layer 2 stamping.

B. IEEE 802.1AS

The development of 802.1AS started as an audio/video technology. The intended use was to support synchronized network streams to be delivered over a LAN and requires time-aware infrastructure. The protocol was defined partially as a strict profile (subset) of 1588v2, but also has extended the 1588 protocol so the compatibility is limited.

IEEE 802.1AS was planned already with a proper protocol layering and from the beginning it was expected to support several network technologies. It has raised the interest of industrial actors and later the audio/video focus was extended to cover any kind of service which needs high precision time synchronization.

The 802.1AS focuses on synchronized services on bridged networks and the standardization is ongoing with high effort. The most important current fields are:

- A frame preemption technique in the bridges (P802.1Qbu)
- An Ethernet traffic scheduling technique (P802.1Qbv)
- New traffic shapers
- Next version of the Stream Reservation Protocol (SRP)(P802.1Qcc)

With IEEE 802.1 TSN, the network itself will use the time that is transported. Both the Ethernet traffic scheduling functionality and time triggered traffic shapers will utilize time for correct execution.

The addition of a reservation protocol, having preemption and scheduling makes this standardization effort very important for industrial actors. This could mean that IntServ-like QoS functions could be deployed on an L2 network, which until now had to accept to have only simple filtering, priorities and very simple traffic management rules.

C. IETF TICTOC

Internet Engineering Task Force (IETF) is working on a time synchronization solution with focus on WANs as well. Their focus is on running time synchronization over large networks, e.g., running 1588 over *Multiprotocol Label Switching* (MPLS).

D. Convergence

It is expected that the two protocols, as they both are being developed by nearly the same actors under the umbrella of the same organization, will be able to at least coexist on the same network. Even more, a cooperation between the protocols would be beneficial, as there 802.1AS could take over in LANs and 1588v3 could cover all the other networks.

V. Architecture possibilities

With a few exceptions, only managed switches are being deployed in industrial Ethernet networks. This is a result of the different requirements, e.g., prioritization and Virtual LANs (VLANs) are rising in the industrial environment compared to the office networks.

A. Discrete switches

Unmanaged switches offer a low-price connectivity solution, where leafs are placed into the same network and the ingress traffic can be treated with the same rules independently of the port. The biggest advantage of the unmanaged devices is their simplicity. The simple silicon offers more deterministic operation (there are no multiple user-configured rules and preferences to check), higher reliability (typically in safety deployments an unmanaged switch represents much lower failure risk), lower heat dissipation and lower price.

Unfortunately, in most cases and despite their advantages, unmanaged switches are not considered as typical requirements in industrial installations. Typical requirements are redundancy functions, traffic prioritization and extended status reports, which are not available in unmanaged devices.

Managed switches offer redundancy and loop-avoidance functions (e.g., RSTP) with logical segmentation using VLANs, remote management with *Simple Network Management Protocol* (SNMP) and troubleshooting features such as port mirroring.

There are also devices in the office networks, located between these two levels, called smart switches. They offer the most of the managed switch functions, but lack for example SNMP management. Introducing a similar class of devices into automation networks where only the necessary protocols are selected might be of interest, since having a more grained approach on switch features can lead to a more cost-effective network architecture. Also, in safety and security fields, it is beneficial to have as simple devices as possible, also in the software running on them. A switch, where a part of the protocols are being left out (e.g., VoIP-related if no VoIP system is in operation) simplifies the checking and certification tasks.

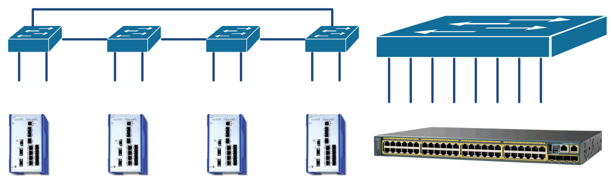


Fig. 4. Typical switch size comparison

There are few arguments against the use of discrete switches and most of them originate from the specific industrial landscape: the low branching factor, which results in a high number of low port-count switches, as shown on Figure 4. The high number of standalone switches and the rugged hardware leads to an expensive network infrastructure with most of the resources being unused.

The low port count is even more apparent in the daisy-chained field networks, where Ethernet is also expected to replace the legacy communication solutions but typically it has to utilize the same topology.

In such environments, using the typical 8-10 port managed switches is rather expensive, as even such a low port count will not be utilized in addition to the higher management effort. To overcome price pressure, excessive engineering complexity and dependency on third party devices, vendors move towards embedded solutions.

VI. Embedded switches

Integrating a switch module into devices like controllers is on the agenda of automation vendors. These modules could take tasks of discrete switches in the lower levels of automation networks. The construction of these units is potentially cheaper than using a separate switch (e.g., a low-end switch fabric) and can provide a few gigabit/second of non-blocking bandwidth.

There are several important issues around the integration of devices. The first is the question of interface towards the host device. The typical architecture offers an internal interface towards the host, which is implemented as a standard, but internal, Ethernet link. This setup is analogue with the discrete switch case, only the interface connecting the host and the switch has been exchanged with the internal connection.

Switch modules, by default, only forward the traffic and all features, which are needed to implement a managed switch, have to be run on the host or the switch module has to be extended with traffic processing capabilities.

Integrated modules are expected to deliver similar performance results as their low port count discrete counterparts and also to offer the similar range of services. The cost of such a solution could be still lower as compared to the separate unit, as several components can be shared with the host, e.g., power supply, casing and user interface, while a potential risk is the software support and compatibility for mixed or brown field deployments.

If the management functions are implemented by using the CPU on the host, multicore platforms can be exploited by moving the forwarding-connected functions to one core and running the other functions on another core, even on a different operating system if needed by utilizing virtualization.

The possible drawback with these modules is that the host is still only connected with one internal port. This means that if the aggregated bandwidth use exceeds the host's bandwidth, the host has no possibility to monitor the whole network traffic. The lack of full monitoring possibility is problematic if the switch module itself cannot report hazardous traffic situations with, e.g., raising a signal when a port experiences traffic congestion. Also the implementation of traffic prioritization can be problematic if not at least parts of the task is implemented in the silicon.

It is clear, that for the industrial applications, unmanaged switches offer less than the required functionality, but on the other hand, fully featured managed switches, especially the low latency ones originally developed for core operation or finance applications are too expensive and most of their features will not be utilized.

A. Minimum acceptable service level

A non-conventional approach is to minimize the implemented features of the devices. Typical requirements state that the switches used should be managed, but the actual feature set is not defined. Currently, managed switches typically implement the whole feature set expected from a managed switch (complying to IEEE 802.1D), but in most cases, only a handful of features are actually used and also out of these, some are enabled by using the default configuration (e.g., weighting of frames in QoS queues).

A reduction in both cost and management effort could be realized by implementing a set of minimum acceptable service-level switches. An office network approach is the smart switch device class, for example the NetGear JFS524e, where the feature set is restricted to offer easier management and cost reduction in hardware. The smart switches represent more a restricted managed switch, but in the industrial environment, an approach from the opposite direction, extending the features of an unmanaged switch might be more interesting.

The motivation to use such devices is partly supported by the reduced development and device cost, but more importantly, the special circumstances of industrial deployments are also supporting this solution. One of the more complex services of the discrete switches is connected to traffic manipulation and security functions.

The gain associated with, e.g., *Internet Group Management Protocol* (IGMP) is that it can reduce the link load by grouping the receivers of multicast streams and also to protect other devices from using resources on traffic, which they have no use for. *Protocols Multiple MAC Registration Protocols* (MMRP) and *Multiple VLAN Registration Protocol* (MVRP), are also expected to reduce traffic load in areas where, e.g., a VLAN has no clients configured.

The other group of protocols are the network operation functions, e.g., RSTP, and IEEE 802.1X using *Remote Authentication Dial In User Service* (RADIUS). These protocols are run to keep the network loop-free and ensure network integrity and to allow secure authentication of new nodes.

Although all of these protocols are useful in an average network topology, in the industry-typical long and sparse trees, their gain is reduced. For increased traffic effectiveness: because of operational safety, networks anyway have to be designed so, that they can carry the whole network traffic, so the gain offered by grouping protocols might be limited. The main problem associated with grouping protocols in the typical line topology is that the resources need to be reserved over the whole path if nodes are expected to join or leave on the ports. The traffic reduction efficiency for line topologies depend on the actual traffic type.

For example, Multiple VLAN Registration Protocol (MVRP) might cut out some VLANs to be carried on a specific path, but if new nodes are allowed to join to a segment, the bandwidth for carrying additional or all of the existing VLANs shall be possible, thus the bandwidth spared by MVRP shall be reserved. In case of multicast protocols, like MMRP can be beneficial, but in this case also, at least the bandwidth need for all multicast groups shall be reserved even if not all of the groups are transmitted.

The execution of RSTP might also be of limited use, if the switches are organized in a chain and in every case, if the main uplink is broken, the other designated port towards the other switches will be chosen. Also, the topology of these networks is very static.

A possible solution is to use a compromise: deploy as simple as possible switches where chained topologies are used and include fully-featured discrete units where a tree connection structure is used (e.g., interconnecting rings or network backbone). Thus, the discrete units can run all the grouping protocols and reduce the load introduced to the ring, but inside the ring no further optimization is done. The simple devices shall be transparent on all protocols they do not support.

VII. Soft switches

Embedded communication solutions are now allowing the implementation of a soft switch processing traffic of several gigabit/s of traffic on low consumption System on a Chips (SoCs). These are typically combined from an embedded CPU, a set of independent network controllers and a chipset, which integrates these into one system.

The positive point with these setups is that the host is the switch: it is possible to monitor the whole traffic flow directly on the interfaces. Also, the platform can provide a good basis for feature extensions toward implementing a router, firewall or network monitoring appliances. The industry can profit here with the evolution of SoCs in the recent years, reaching very high performance and integration with low power consumption and versatile usage areas, e.g., different ARM cores with integrated analogue-digital converters, various network adapters, on dye memory, cryptographic functions and storage.

A high performance, multicore SoC can also serve as a platform for automation tasks and with the use of a multicore CPU, the communication and automation tasks could be run separated. With virtualization being also offered in this price and performance segment, it is also possible to even run tasks on separate, isolated operating systems, using dedicated cores.

The main drawback of soft switches is the absence of the dedicated switching fabric. The throughput of the platform is prone to the actual implementation of the system and network adapter interconnection, used drivers and operating systems as well. Also, the limitations of the bus system and the network interfaces are summed, which can lead to insufficient performance in low latency environments. The price tag of such a solution can be justified if the device is utilized also in other tasks not only bridging.

VIII. Feature comparison

The reviewed architectures show that if the switching solution is chosen, the future possibilities regarding traffic management, performance and feature set are being reduced.

Discrete switches offer high performance and a long list of management features and supported protocols. Embedded switch modules are implementing switching, but protocol and management features have to be implemented by the host or by a separate CPU and they only offer statistic multiplexing towards the host if utilized bandwidth exceeds what the host interface can carry. Soft switches are in practice implementing the embedded switch scenario but without the hardware switch module, thus while offering full access to all traffic crossing the interfaces, they also suffer from the largest delays.

From the forwarding performance side, for large port counts, discrete switches offer the best solution, since a high-speed, non-blocking backplane is a hard requirement in this area. For smaller and medium sized switches (4-16 ports), an integrated module can also be viable. For low port count even the cheaper backplane solutions can provide enough bandwidth. It is also less probable, that such a switch will be experiencing a situation where all of the ports are fully utilized.

Our measurements on the forwarding latency and throughput of switches showed marginal differences between discrete and embedded solutions while the tests executed on the soft switch platform resulted in weaker performance figures.

IX. Performance Measurement

A. *Measurement and test equipment*

Our test was implemented with the use of an array of discrete managed switches. The traffic generator was a Softing Industrial Ethernet Tester (OEM Psiber LanExpert 80), which can generate traffic between its two gigabit Ethernet interfaces and was acting as traffic source and sink.

Our tests were split into two areas: one was to measure the latency between two ports of the same switch to provide a way to compare the raw performance. The second area was to show switched Ethernet behavior in a typical industrial setup, where switches are chained and the ingress and egress links are only 100Mbps while inter-switch links are 1Gbps. The initial results based on the LanExpert measurements showed no significant difference in latency or throughput between the embedded and discrete units.

To measure the latency between 100Mbps endpoints, we need preciseness ideally at the level of a bit-duration or better, which is 10ns for the Fast Ethernet. Since the LanExpert's measurement capabilities were not satisfactory for generation of the statistics and exact measurement of forwarding behavior in a cascade, we decided to use the EtherCAT network consisting of the master (a Freescale P2020 development board utilizing a dual-core PowerPC e500 CPU) and two slaves (Figure 5. EtherCAT provides service called *Distributed Clock* (DC), which can precisely synchronize clock in slaves with time resolution of 10ns and has dedicated hardware in slaves to measure network latency.

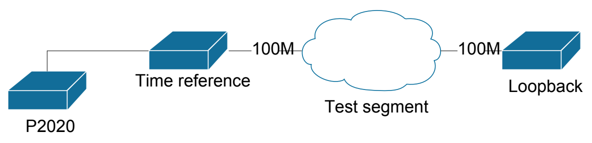


Fig. 5. Testbed setup

To assess the performance of the selected equipment and to be able to provide guidelines for network planning, we set up the following measurements.

B. Default forwarding latency

Measurement of the time it takes for the frame to traverse the switch. It is composed from store and forward latency (L_{SF}), the switch fabric latency (L_{SW}), the wire line latency (L_{WL}) and the queuing latency (L_Q) [24].

L_{SF} depends on the frame length. The results are expected to show a linear growth of the latency with the longer frames [25].

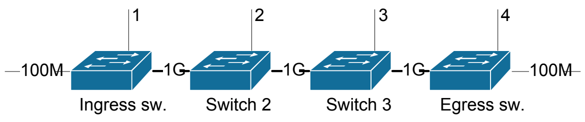


Fig. 6. Test segment setup

Our architecture related measurement scenarios deals with latency between endpoints (both with 100Mbps) of serial connected switches and without any additional interfering traffic (see Figure 6). The purpose is to see the raw latency scaling of a network built by a chain of switches. We have four scenarios, each of them consisting with 1 up to 4 switches. Switches are between themselves connected with 1Gbps link. Initial measurements showed, in accordance with the LanExpert measurements, no significant difference between the discrete and embedded units, so the testbed was created by using 4 RuggedCom RS940G switches.

C. Standalone forwarding

Latencies and throughput between two interfaces of the same switch was measured with the LanExpert device and the results showed no significant difference between the capabilities of the embedded or the discrete units.

Measurements were performed on switches, which represent a significant part of the market: RuggedCom RS940G, Hirschmann RSR30, Moxa EDS-G509, a board based on Marvell 88E6352 switch chip and a soft switch using a stock Ubuntu linux and an Intel Xeon CPU with four chipset-integrated gigabit Ethernet interfaces. As a control, a test was also executed on a Cisco SG 200 switch (approximately the same performance class as the tested industrial variants), where differences in the results were also insignificant compared to the industrials. The measured latencies of the Marvell module are marginally lower, than the discrete counterparts, which are expected to be the result of the simpler architecture, as the module in the tested form implements only an unmanaged switch.

The only considerable difference could be observed with the soft switch platform. It was not expected to hold the same latency figures but the maximal frame frequency of approximately 180kfps is low compared to the rest of the devices (Table I). Although the latency is also higher (Table II), the figures stay mostly within acceptable range for the majority of networking tasks. The low throughput observed with shorter frames in contrast, limits the specific setup's usability since it will not be able to utilize the bandwidth in case of a setup like our test segment, where two interfaces need to carry the aggregated traffic. It is expected that with different operating system and driver optimizations, better performance can be achieved.

TABLE I
THROUGHPUT IN K FRAMES PER SECOND FOR RESPECTIVE FRAME SIZES USING 1 GBPS LINKS

Frame size	RS940G	RSR30	EDS-G509	88E6352	soft
64	1481	1485	1485	1485	179
128	840	842	842	842	178
256	452	452	452	452	178
512	234	234	234	234	166
1024	119	119	119	119	119
1280	96	96	96	96	96
1518	81	81	81	81	81

TABLE II
LATENCY IN MICROSECONDS FOR RESPECTIVE FRAME SIZES USING 1 GBPS LINKS

Frame size	RS940G	RSR30	EDS-G509	88E6352	soft
64	5	5	5	3	16
128	5	5	5	4	16
256	6	6	6	5	18
512	8	9	8	7	33
1024	12	13	12	11	114
1280	14	15	14	13	93
1518	16	17	16	15	99

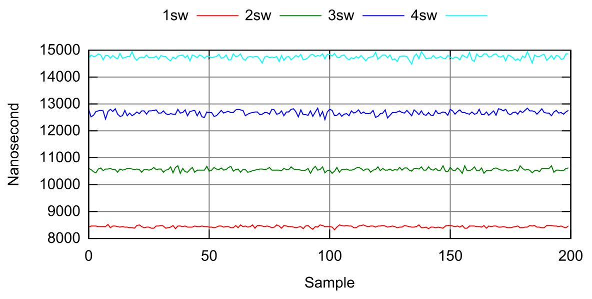


Fig. 7. Scenarios 1-4

D. Scaling of latency in a chain

Our measurements using the testbed extended with a variable cascade of switches (Figure 6 show that the discrete switches scaled as expected. When no additional traffic was injected to the measured interfaces, the latency growth was linear with minor variations. Measurements using an industry-typical scenario with 100 Mbps edge links and 1 Gbps internal links were executed. Four scenarios were measured, compromising of chains of 1-4 switches (Figure 7).

Also, the histogram on Figure 8 shows the expected behavior: the longer the chain is built, the wider is the range of latencies measured. The determinism of the switching solutions can be seen on the measurements and that in low traffic installations a linear growth of latency can be expected.

The histogram of the measurements showed the expected result, with having the most step-like distribution at using one switch and a still narrow but wider distribution of frame latencies for longer switch chains.

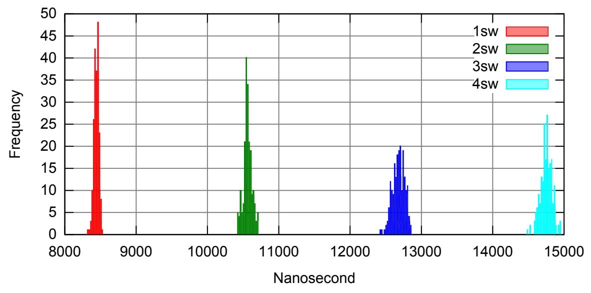


Fig. 8. Histogram Scenarios 1-4

E. Behavior under load

As an extension, the testbed was also used to execute measurements under load, to check how the devices react on a selection of traffic situations.

Behavior on crossing traffic was tested, where the switch fabric was tested if it is in fact non-blocking. In the range of devices tested, there was no significant difference compared to the no traffic scenarios (Figure 9, which shows that the hardware is using a non-blocking architecture.

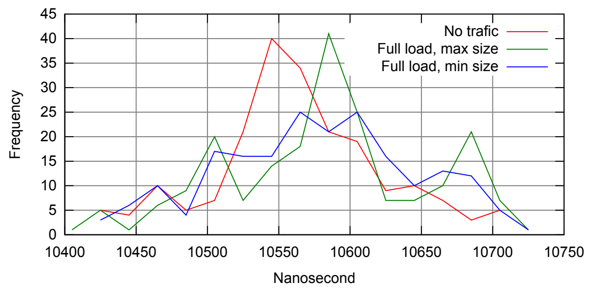


Fig. 9. Switch fabric stress test histogram

Also when jitter with regard to frame length was checked, the results were in line with the expectations with showing stable low deviations in the no traffic case (where the measurement frame was the only payload transmitted on the network) and the case where minimum length frames were sent over the network.

The traffic made from full length frames, as expected were suffering more (Figure 10), as both their transmission time and the possible waiting time is longer.

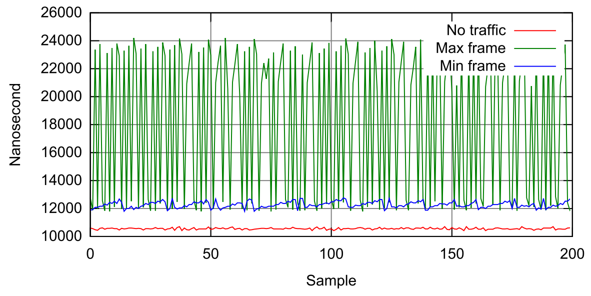


Fig. 10. Jitter with different frame sizes

The histogram (Figure 11) also shows the expected behavior, where in the no other traffic situation the roundtrip times have little variation, where at the maximum length frame measurements show a relatively high jitter, which is introduced by the long transmission time.

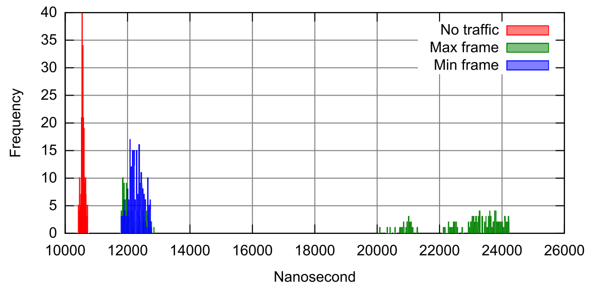


Fig. 11. Histogram of jitter for different frame sizes

One of the main sources of industrial skepticism against Ethernet is the non-deterministic operation of the network. While our experiments show that the expected non-deterministic transmission times do happen and that in case of long frames, the jitter is high compared to the minimal or the typical transmission times, it is important to point out, that this low jitter values only have relevance in a few applications only. The measurements showed a maximum deviance of approximately 5000 nanoseconds per hop. In more realistic traffic mixes, where the industry-typical short to medium sized frames would take the bulk of the traffic, also with the typically low bandwidth utilization, the experienced jitter would be much lower. Still, even if the worst case with maximum length frames is taken, compared to real applications, like process automation and most of manufacturing, the tolerance of the measurement or control loop is several magnitudes higher than the jitter introduced by the communication subsystem.

The insignificance of jitter is especially valid if there is a Layer 3 device in the path (e.g., firewall or a gateway), where it is expected, that device alone will introduce a higher delay than the whole section of L2 network before and after. In case operator reaction is needed, the tolerance is even higher, as humans react in fraction of a second or close to a second, so the delay introduced by the operator would be much higher than the jitter.

X. Outlook

In the foreseeable future, industrial network installations will follow the chained structures, mainly because of the industry's traditional skepticism for changes and partly because of the costs associated with additional cabling.

The evolution of industrial Ethernet will proceed towards higher device integration, where first, the switches will be included in the automation controllers and other devices, and which will proceed with integrating most of the active network devices.

It is expected, that only the site level high performance devices, which have a large resource need on their own will still be independent devices, but these will most probably be shared between the industrial and telecom fields. This sharing is opening a very promising field, *Software Defined Networking* (SDN) for the industrial market as the support functions will be available in the high-end active devices supplied by networking companies and the industrial area offers a relatively simple, static and homogeneous world.

The strongly limited industrial environment represents a good starting point for deploying networks using SDN. The range of devices is much smaller as compared to office environments and, especially in green field deployments; the network is controlled by one vendor.

Traffic on an industrial network also tends to be deterministic, having periodic communication as part of the control loops, periodic updates of the logs and infrequent best effort traffic related to end node maintenance.

SDN fits very well the wishes of industrial vendors:

- hiding network complexity, which lowers the pressure on engineers,
- allows simpler service deployment,
- possibility for central resource management,
- seamless integration of wireless technologies.

A. Network complexity

The network topology and node types provide a good basis for experimenting with SDN in an industrial setting. The used topologies are not complex, the type of nodes is limited and the typical operation is static.

So an abstraction from the physical nodes could be implemented with limited effort as there is no need for a generic solution. In a typical installation, even before using integrated switches, a handful of types would be deployed together with a large number of end nodes, which from the network's viewpoint, operate nearly the same. As the typical traffic is machine-to-machine, the dynamism of the traffic is limited compared to non-industrial scenarios.

The use of SDN would although come from a different motivation as compared to, e.g., telecom installations. While there the use of SDN would enable better dynamism and abstraction of control from forwarding to allow more flexible network utilization, the industrial scenario is more about having a centralized control over the network, including forwarding decisions (no local configuration of the integrated devices is necessary) and for implementing a type of call admission control. With admission control, the SDN controller could check if a newly deployed control loop or other traffic source could be securely fit into the current resource utilization or there is a need for more changes in the infrastructure.

Also the granularity of control would be extended, as with SDN it is possible to influence forwarding decisions on a per flow basis. This means, that the engineer will be able to globally prioritize traffic flows, have a total overview on forwarding decisions and have a central entity to ensure that the configuration of devices is actually the one what is expected.

B. Service deployment

In and SDN case, the central management entity can change the configuration and forwarding behavior of the underlying devices. This could lead to cost savings in both infrastructure, as currently over provisioning is typical in the field, or in lowering the resources needed for engineering since the traffic estimates would be made by the SDN system and not the engineers.

An additional gain would be that an SDN system could deploy a new service without disturbing the current operation, which would reduce costs related to planned downtimes. The abstraction of hardware and central intelligence leads also towards simpler integrated devices.

C. Central resource management

Currently, SNMP-based Network Management Systems (NMSs) are widely used for monitoring the health and status of large network deployments. Using SDN could also here be beneficial, as the monitoring functionality would be extended with the ability of actively changing configurations and resource allocations if needed.

D. Wireless integration

Another key field currently is the integration of wireless networks into industrial deployments. Although there is skepticism for the usage of wireless solutions both because of determinism and security, already now, a growing part of the deployments use wireless solutions. Here SDN could help with integration of wireless technologies by checking if the needs of a new service, e.g., can be satisfied with a path having one or more wireless hops or a new rule has to be deployed into the network to steer the traffic of that service on a different path.

E. Vendor-neutrality

As also Ethernet deployments are getting old enough to be part of plant upgrades or new deployments with multiple suppliers, SDNs capability of giving a vendor-neutral abstraction of the network would be a great advantage.

XI. Conclusion

Our review shows that with regard to forwarding performance and latency, embedded switching solutions present a competitive solution compared to discrete units. Although, the offered set of features might differ and for managed functions, either the host CPU or an additional CPU for the switching board needs to be used, the performance expectation can be the same.

Soft switching on the other hand might be problematic when using a non-real time operating system and non-optimized drivers. If the software selection would move towards these, on the other hand, the flexibility of the platform would be limited. Our measurements showed that soft switches might be too slow to be used in a chained topology, but might be applicable in cases, where additional processing is required, for example as a controller with several network interfaces.

Our conclusion is that if there are no clear requirements for traffic monitoring capabilities exceeding the bandwidth of the host-switch module link, embedded switches are a viable and effective solution for low branching factor industrial networks.

Soft switches are a viable solution for implementing routers or other network functions, where the additional latency compared to the other switching solutions is not critical as the processing of the data on higher layers will contribute to more latency and jitter as the switching.

In a broader perspective, the identified problems for industrial Ethernet switching lie more on the feature set than in the hardware performance.

References

[1] G. Kálmán, D. Orfanus, and R. Hussain, "An overview of switching solutions for wired industrial ethernet," in *Proceedings of the Thirteenth International Conference on Networks*, IARIA, Nice, France, 2014, pp. 131–135.

[2] J. Kay, R. Entzminger, and D. Mazur, "Industrial ethernet- overview and best practices," in *Pulp and Paper Industry Technical Conference, Conference Record of 2014 Annual*, June 2014, pp. 18–27.

[3] F. Cen, T. Xing, and K.-T. Wu, "Real-time performance evaluation of line topology switched ethernet," *International Journal on Automation and Computing*, vol. 5, no. 4, pp. 376–380, October 2008.

[4] J. Jasperneite and P. Neumann, "Switched ethernet for factory communication," in *Emerging Technologies and Factory Automation, 2001. Proceedings. 2001 8th IEEE International Conference on*, October 2001, pp. 205–212, vol.1.

[5] J.-P. Georges, N. Krommenacker, T. Divoux, and E. Rondeau, "A design process of switched ethernet architectures according to real-time application constraints," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 3, pp. 335 – 344, 2006.

[6] N. Kakanakov, M. Shopov, G. Spasov, and H. Hristev, "Performance evaluation of switched ethernet as communication media in controller networks," in *Proceedings of the 2007 International Conference on Computer Systems and Technologies*, ACM, 2007, pp. 30:1–30:6.

[7] J.-P. Georges, T. Divoux, and E. Rondeau, "Comparison of switched ethernet architectures models," in *Emerging Technologies and Factory Automation, 2003. Proceedings. ETFA '03. IEEE Conference*, vol. 1, September 2003, pp. 375–382 vol.1.

[8] G. Marsal, B. Denis, J.-M. Faure, and G. Frey, "Evaluation of response time in ethernet-based automation systems," in *Factory Communication Systems, 2006 IEEE International Workshop on*, 2006, pp. 95–98.

[9] H.-T. Lim, L. Volker, and D. Herrscher, "Challenges in a future ip/ethernet-based in-car network for real-time applications," in *Design Automation Conference (DAC), 2011 48th ACM/EDAC/IEEE*, June 2011, pp. 7–12.

[10] T. Skeie, S. Johannessen, and O. Holmeide, "Timeliness of real-time ip communication in switched industrial ethernet networks," *Industrial Informatics, IEEE Transactions on*, vol. 2, no. 1, pp. 25–39, February 2006.

[11] J. Jasperneite, M. Schumacher, and K. Weber, "Limits of increasing the performance of industrial ethernet protocols," in *Emerging Technologies and Factory Automation, 2007. ETFA. IEEE Conference on*, September 2007, pp. 17–24.

[12] A. Manita, F. Simonot, and Y.-Q. Song, "Multi-dimensional markov model for performance evaluation of an ethernet switch," INRIA, Tech. Rep. 4813, 2003.

[13] K. Beretis and I. Symeonidis, "Experimental evaluation of end-to-end delay in switched Ethernet application in the automotive domain," in *SAFECOMP 2013 - Workshop CARS (2nd Workshop on Critical Automotive applications : Robustness & Safety) of the 32nd International Conference on Computer Safety, Reliability and Security*, M. ROY, Ed., Toulouse, France, 2013, p. NA.

[14] J.-D. Decotignie, "Ethernet-based real-time and industrial communications," *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1102–1117, June 2005.

[15] A. Jacobs, J. Wernicke, S. Oral, B. Gordon, and A. George, "Experimental characterization of qos in commercial ethernet switches for statistically bounded latency in aircraft networks," in *Local Computer Networks, 2004. 29th Annual IEEE International Conference on*, November 2004, pp. 190–197.

[16] H. Systems, "Real time services (qos) in ethernet based industrial automation networks," Hirschmann, Tech. Rep., 2000.

[17] M. Knezic, B. Dokic, and Z. Ivanovic, "Performance evaluation of the switched ethercat networks with vlan tagging," *Serbian Journal of Electrical Engineering*, vol. 9, no. 1, pp. 33–42, 2012.

[18] M. Kezunovic, A. Sprintson, J. Ren, and Y. Guan, "Signal processing, communication, and networking requirements for synchrophasor systems," in *Signal Processing Advances in Wireless Communications (SPAWC), 2012 IEEE 13th International Workshop on*, June 2012, pp. 464–468.

[19] *"IEEE Standard Profile for Use of IEEE 1588 Precision Time Protocol in Power System Applications*, IEEE Std. C37.238-2011, 2011.

[20] A. Mahmood, R. Exel, and T. Sauter, "Delay and jitter characterization for software-based clock synchronization over wlan using ptp," *Industrial Informatics, IEEE Transactions on*, vol. 10, no. 2, pp. 1198–1206, May 2014.

[21] Siemens, "Profinet answers for industry," Siemens, Tech. Rep., 2010.

[22] *1588-2008 IEEE Standard for a Precision Clock Synchronization Protocol for Networked Measurement and Control Systems*, IEEE Std., 2008.

[23] F.-J. Goetz, "IEEE 802.1AS bt (gPTP) & IEEE 1588 v3 (PTP v3)," 2013, presentation of Siemens at IEEE 802.1 TSN WG Meeting.

[24] S. RuggedCom, "Latency on a switched ethernet network," RuggedCom, Tech. Rep., 2008.

[25] J.-P. Georges, T. Divoux, and E. Rondeau, "Network calculus: Application to switched real-time networking," in *Proceedings of the 5th International ICST Conference on Performance Evaluation Methodologies and Tools*, ser. VALUETOOLS '11. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2011, pp. 399–407.

Management System Scalability Evaluation in Multi-domain Content Aware Networks

Eugen Borcoci, Mihai Constantinescu, and Marius-Constantin Vochin
University POLITEHNICA of Bucharest
Bucharest, Romania
emails: eugen.borcoci@elcom.pub.ro
mihai.constantinescu@elcom.pub.ro, marius.vochin@elcom.pub.ro

Abstract — **This paper studies and evaluates the scalability properties of a resource management subsystem as a component of a networked media eco-system. The overall system aims to offer multimedia delivery with configurable guarantees of Quality of Services, over multi-domain networks, to large communities of users. The transport infrastructure is based on creation of data plane logical slices named Virtual Content Aware Networks. These slices are realized under control of a management plane, comprising centralized per-network domain "controllers", cooperating to construct parallel data planes, which span multiple IP network domains independently managed. In this multi-domain context, the management system scalability properties are important, while the problem is similar to the multi-controller inter-communication in emerging multi-controller Software Defined Networking technologies. The management system architecture considered in this paper has been previously defined. This work additionally provides a simulation model and results, concerning the scalability of the multi-controller communications subsystem. It is shown that the proposed control approach is conveniently feasible in a multi-domain network environment.**

Keywords — Content-Aware Networking; Software Defined Networking, Multi-domain; Management; Resource provisioning; Future Internet.

I. INTRODUCTION

Many novel architectural solutions are proposed for the Future Internet and these are still open issues for research. Their major target is to solve some of the recognized fundamental architectural limitations of the traditional Internet, and reduce its ossification, while answering better to the current challenges related to new services needs and also offering a better support for the Internet global extension [1][2][3].

A significant trend recognized in the current and also estimated for the future Internet is a strong information/ content-centric orientation, including media content distribution. Consequently, changes in high level services and networking have been recently proposed, including modifications of the basic architectural principles. Some "revolutionary" approaches are often referred to, as *Information Centric Networking (ICN),* or equivalent, *Content Oriented/Centric Networking (CON/CCN)* [4][5].

In parallel, evolutionary (or incremental) solution emerged, introducing Content-Awareness at Network layer (CAN) and Network-Awareness at Applications layers (NAA). This approach increases the amount of information on the flows transported at network level (compared to the content-agnostic IP) and provides summary information about transport characteristics to the application/services layers. Thus, a powerful cross-layer optimisation loop can be created between the transport on one side and applications and services on the other side.

An "orthogonal" new trend, targeting to achieve more flexibility and programmability in networking is the *Software Defined Networking* (SDN) architecture and its associated OpenFlow protocol [6][7][8]. In SDN the control plane and data planes are decoupled and the network intelligence is more centralized in so-called SDN controllers, thus offering possibilities of enhanced management and also flexible/programmable control of the resources. However, applying the SDN centralized control in a wide area network (WAN) context does not scale [9]. Several controllers are required, each one controlling a limited network region. In order to get an overall logic view upon the whole network, it is necessary an inter-controller communication subsystem. The scalability of this is an open research issue, and is also studied in this work.

The European FP7 ICT research project, *"Media Ecosystem Deployment Through Ubiquitous Content-Aware Network Environments"*, ALICANTE [10], adopted the NAA/CAN approach, to define, design, and implement a Media Delivery Ecosystem, spanning multiple network domains.

This work considers as a basis the ALICANTE management architecture [10][11], which is, conceptually, partially similar to a multi-controller SDN architecture, with respect to the distribution of the main management and control functions among several controllers. Communication between controllers is necessary in order to accomplish multi-domain tasks.

This paper is *a continuation and extension of the work presented in* [1]. It has extended the scenarios to study the scalability aspects of the ALICANTE multi-controller signaling subsystem (where each controller independently allocates transport resources in its managed domain). The simulation approach has been based on the *Extended Finite State Machines* (EFSM) model [12]. Note that the main task of the studied subsystem is to provide resources for a multi-domain Virtual Content Aware Network (VCAN) constructed at request of a Service Provider (SP) entity, which will be the actual VCAN "user".

We recall that scalability of a system (in case that this system can be abstracted to a graph) can be seen from two points of view: horizontal and vertical scaling. Scaling *horizontally* (or *scaling out*) means that the system allows addition of more "nodes" to it (e.g., adding a new computer to a distributed software application), while still preserving an approximate linear increase in cost and complexity. Scaling *vertically* (or *scaling up*) means that the system allows addition of resources to a single node in a system (e.g., addition of CPUs or memory to a single node). *In this study, the target is the horizontal scalability properties of the inter-controller communication subsystem.*

Section II presents samples of related work. Section III shortly describes the multiple-domain resource management architecture considered in this study. Section IV defines the inter-controller communication subsystem. Section V introduces the simulation model to study the signaling scalability and Section VI presents samples of the simulation results. Conclusion, open issues, and future work are shortly outlined in Section VII.

II. RELATED WORK

The ICN/CCN/CON approaches are very promising for the future Internet [2][3][4][5]. However, they raise some research and especially deployment challenges, given novel paradigms proposed, concerning naming and addressing, content-based routing and forwarding, management and control framework, in-network caching, etc. Other issues are related to scalability, especially related to the amount of processing and resources in the ICN routers and also security. ICN/CCN/CON technologies require significant changes of the current Internet deployments and protocols.

That is why, some evolutionary (incremental) approaches have been proposed such as CAN/NAA, built as overlays upon existing Internet infrastructures. These evolutionary solutions are (hopefully) able to support in a better way the seamless development of the networked media systems and also the market orientation towards content, while paving the road to full ICN/CCN.

Content Delivery Networks (CDN) [13], using large physical and logical infrastructures, provide improved content related services based on content replication. They offer fast and reliable applications and services by distributing content to cache or edge servers located close to users. A CDN is a collection of network elements arranged for more effective delivery of content to end-users. The typical functionality of a CDN includes [13]: content outsourcing and distribution services to replicate and/or to cache content to distributed surrogate servers, on behalf of the origin server; request redirection and content delivery services to direct a request to the closest suitable surrogate server; content negotiation services to meet individual user needs; management services for network components, accounting, and to monitor and report on content usage. Different from ICN/CCN, the CDNs can use the current Internet support; they are largely developed in the real world. However the CDNs need a large number of powerful servers and should apply sophisticated procedures related to content distribution and caching policies.

In SDN [6][7][8][14], the main network intelligence is centralized in SDN controllers. Such an approach offers a better and flexible control of the resources, quality of services, etc., due to the possibility to have an integrated/overall knowledge of the network capabilities in the control plane and by allowing programmability of the network resources (forwarding plane). To this aim, standardized south-bound (between controller and forwarding devices – e.g., like OpenFlow) and north-bound interfaces (API for applications) are currently developed for the SDN "layer". Therefore, operators and service providers will get more freedom and gain speed in developing their services, without waiting long time for new releases of vendor's networking equipment.

Although SDN technology seems to be very attractive, e.g., for data centers but also for core wide area networks, it exposes also many research challenges and open issues, both from architectural and from deployment point of view. The degree of centralization and relationship with scalability and reliability are examples. An extension of the SDN concepts is proposed in so-called *Software Defined (Internet) Architecture* [14], where the idea is to decouple the architecture from infrastructure, aiming to lower the barriers to architectural evolution. The SDIA approach tries to exploit SDN concepts but also traditional technologies (e.g., Multi Protocol Label Switching MPLS, software forwarding-performed on edge routers, etc.) in order to obtain evolvable architectures. SDN and SDIA are still evolutionary in contrast with those technologies called "clean slate" - like ICN/CCN, which are architecturally disruptive.

Currently, there exist concerns about SDN's performance, scalability, and resiliency [7][9][15], the main source for these problems being the centralization concept. It is clear that a central controller will have a limited processing capacity and the solution will not scale as the network grows (increased number of switches, flows, high bandwidth needs, etc.). The controller's performance can be increased, but a second solution is to define a SDN multi-controller architecture. However, SDN still has as objective to get a consistent centralized logical view upon the network; this creates a need for controllers *to cooperate and synchronize their data bases,* in order to provide together a consistent view at network level. Work in progress is developed at IETF towards defining an inter-controller communication system [16]. While, with respect of the vertical protocols between Control and Data Plane we have seen significant progress in specifying Open Flow versions [8], and implementing several types of controllers [7][15], the inter-controller cooperation and scalability issues are still under research.

In order to identify the degree of interest for the problems to be studied in this work, we shortly describe below the relationship of the approach considered here, to the above solutions (ICN/CDN/SDN). The full architecture has been defined in ALICANTE project [10], being a mid-way solution among ICN/CDN/SDN, given that:

- it is an architecture CAN/NAA oriented, overlapping with ICN/CCN in the sense that some degree of content awareness exists at the network element level; due to

this, different levels of QoS guarantees can be offered to the users;
- it has CDN similarities, given that it is oriented to content delivery with QoS guarantees; however it is cheaper, given that it does not need necessarily to replicate content in many caching servers;
- it adopted (for the multi-domain management) the SDN partial centralization concepts where several controllers exist, each managing a region and cooperating with others. In such a way, in our system, an overall image of the multi-domain resources always exists at management level, similar as in SDN.

In particular, the work of this paper is focused on the scalability of the inter-controllers communication protocols, when the number of the involved network domains (to support a requested VCAN) is variable.

III. Multi-domain Management System Architecture

Additional details of the ALICANTE architecture aside of the multi-domain resource managers (controllers) can be found in [10], [11], [17].

In this architecture, several cooperating *environments* are defined, containing business entities/actors:
- User Environment (UE), containing the End-Users;
- Service Environment (SE), containing Service Providers (SP) and Content Providers (CP);
- Network Environment (NE), where we find a novel business entity called CAN Provider and also the traditional Network Providers (NP) managing the network elements, in the traditional way at IP level.

The “environment” is defined as a generic grouping of functions, working for a common goal and, which, possibly, might vertically span one or more several architectural (sub-) layers.

In the Data Plane the logically isolated VCANs are realized as parallel logical data planes, constructed by optimising inter and intra-domain mapping of VCANs, onto several domain network resources. The content awareness is realized by special edge routers called Media Aware Network Elements (MANE) while inside the network domain regular core routers are used. An early design decision has been adopted: VCANs are limited to core networks; they do not span the access and local networks. The reason is related to high variability in edge networks technologies and large variation of the methods to control the connectivity resources in the first and last mile segments of an end to end (E2E) chain.

In the following text of this paper only the management and control plane issues will be discussed as they are the main objectives of this study.

Dynamic (negotiation-based) Service Level Agreements (SLA) can be established between actors. Using SLAs, several Service Providers can independently ask, to a CAN Provider to perform the required resources provisioning for customizable Virtual Content Aware Networks, and then SPs may use them for media flow transport. Network Providers can cooperate to VCAN construction but they still preserve their independency in terms of their own resource allocation for each VCAN requested and constructed. Flexible connectivity services have been achieved in terms of QoS, offering: Fully/partially/un- managed services [17].

The architecture supports both vertical and horizontal integration in terms of SLAs, to offer (edge-to-edge) several levels of guarantees. A partially distributed Management and Control (M&C) plane exists – where each domain has its own Intra-domain Network Resource Manager (Intra-NRM) and an associated CAN Manager). This structure supports all actions for large scale provisioning.

The VCANs are flexible in the sense that they can support (simultaneously) several communication modes: unicast, multicast, broadcast, P2P and combinations with different levels of QoS/QoE, availability, etc. [18]. End Users and some residential gateways called here Home-Boxes can ultimately benefit from CAN/NAA features by using VCANs.

Services Providers are not burdened with tasks to construct the VCANs; they simply ask VCANs (with some characteristics - topology, capacities of traffic trunks, QoS classes, etc.) to be provisioned by a CAN Provider. In case of success of the SLA negotiation between SP and CANP, the VCAN configurations are installed in the routers and then the Service Provider can use the customized connectivity services. The architecture assures QoS and Quality of Experience (QoE) optimization based on: CAN/NAA interaction; cooperation between resource provisioning (based on SLA agreement) and media flow adaptation; hierarchical monitoring at CAN and network layers cooperating with the upper layers. Apart from its focus on media flow delivery with guaranteed QoS, the ALICANTE architecture is enough general as to transport also non-characterized flows of traffic, in “best effort” style. In this way the architecture provides an answer to “network neutrality” requirements.

The simplified ALICANTE VCAN management architecture is presented in Figure 1. The picture only presents the Service Environment (instantiated here by the Service Provider) and Network Environment (the assembly of the bottom blocks). The User Environment is not represented, given that it is not directly involved in VCAN management.

The management and control architecture is conceptually similar to SDN, although not following full SDN specifications (actually the ALICANTE architecture does not use an OpenFlow like protocol). Both architectures are evolutionary and can be seamlessly developed. The Control Plane and Data Plane are separated. Note that Control Plane in SDN terminology is here actually Management and Control Plane. The functionalities like QoS constrained routing, resource allocation, admission control and VCAN mapping are included in the CAN Manager.

The “virtualization” of the network is performed by Intra-domain Network Resources Managers (Intra-NRM), which hides the characteristics of MPLS technology by

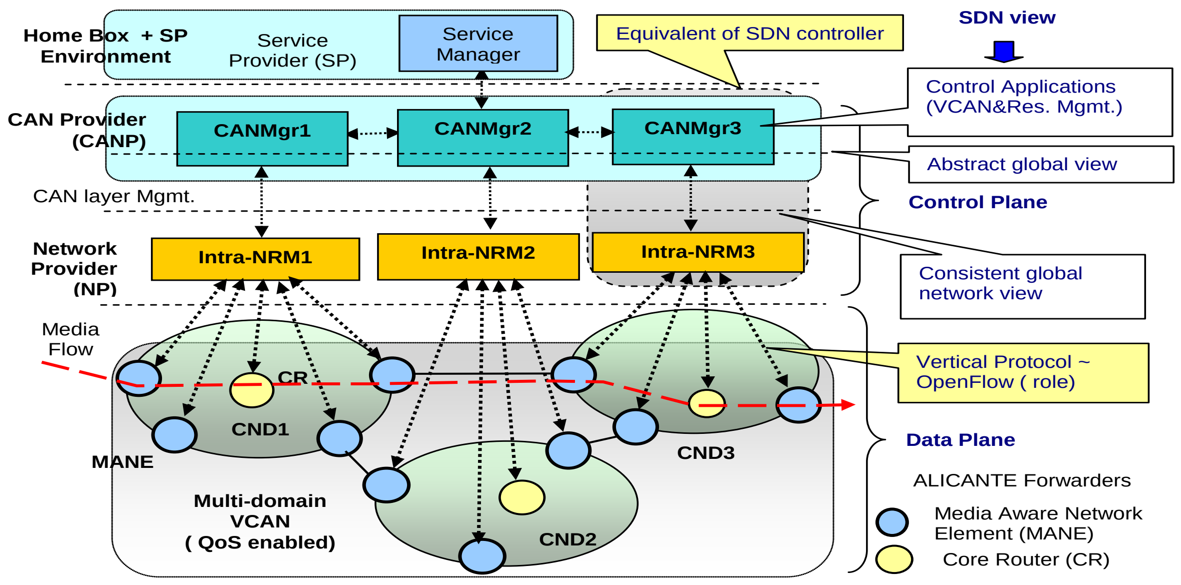


Figure 1. ALICANTE partially centralized management architecture and equivalence with SDN
Notations: Service Environment: SP – Service Provider; Network Environment: CANP - CAN Provider; NP – Network Provider; CND - Core Network Domain; CANMgr - Content Aware Network Manager; Intra-NRM – Intra-domain Network Resource Manager; MANE – Media Aware Network Element

delivering to the CAN Managers an image of abstract matrix of connectivity logical pipes.

In our case the [*CAN Manager* + *Intra-domain Network Resource Manager*] play together the role of an SDN controller for a network domain, controlling the Media Aware Network Elements (MANE) edge routers and interior regular core routers. Actually, we have a multi-domain logical network governed by several "SDN controllers", – which cooperate for resource management and routing. However, the degree of centralization is configurable in ALICANTE by defining any placement of CAN Managers, the network regions and sets of routers to be controlled.

In both SDN and this architecture the Control Plane software is executed on general purpose hardware. The decoupling of the control with respect to specific networking hardware is realized in the sense that MANE, core routers and network links are viewed by the upper CAN layer in abstract way.

The Data Plane is programmable: all configurations for MANE and Core routers are determined at CAN level by Management and Control (M&C) and then downloaded in the routers. ALICANTE architecture defines the control for a whole network (and not for single network devices): at CAN Manager level there exists an overall image on the static and dynamic characteristics of all VCANs; at Intra-NRM level there is a full control on the network domain associated with that Intra-NRM.

In SDN and our case also, the network appears to the applications and policy engines as a single logical switch. In our case, the network appears at higher layers as a *set of parallel planes VCANs*. This simplified network abstraction can be efficiently programmed, given that the VCANs are seen at abstract way; they can be planned and provisioned independently of the network technology.

IV. Inter-domain Management Communications

One CAN Manager (belonging to CAN Provider) is the initiator of VCAN construction, at request of a Service Provider. The VCANs asked should be mapped onto real multi-domain network topology, while respecting some QoS constraints. This provisioning is done through negotiations [11], performed between the initiator CAN Manager and CAN Managers associated to each network domain. In other words, if necessary, the initiator communicates with other CAN Managers, to finally agree: first, transport resources reservation and then a real allocation (i.e., installation in the network routers) of network resources necessary for a VCAN.

A CAN planning entity inside each CANMgr runs a combined algorithm, performing QoS constrained routing, VCAN mapping and logical resource reservation. In this set of actions, it is supposed that the initiator CANMgr knows the inter-domain topology at an overlay level and also a summary of each network domain topology, in terms of abstract trunks (e.g., *{ingress, egress, bandwidth, QoS class, …}*). This knowledge is delivered by an additional discovery service, whose description is out of scope of this paper. Previous paper [17] has proposed and presented the development and implementation of the combined VCAN mapping algorithm.

The overall system flexibility and scalability essentially depends on its Management and Control. For VCAN planning, provisioning and exploitation, it was adopted per-domain partially centralized solution; this avoids full-centralized VCAN management (non-scalable), but allowing a coherent per-domain management. However, the initiator CAN Manager (like in SDN approach), has the overall consistent image of a multi-domain VCAN.

There is *no per-flow signaling* between CAN Managers. The VCAN related negotiation between SP and CANP (concluded by a SLA) is performed per each VCAN, described in terms of topology and aggregated traffic trunks. The SP negotiates its VCAN(s) with a single CAN Manager irrespective, if it wants a single or a multi-domain spanned VCAN.

An hierarchical overlay solution is applied for inter-domain peering and routing [18], where each CAN Manager knows its inter-domain connections. The CAN Manager initiating a multi-domain VCAN is the coordinator of this hierarchy. It knows the inter-domain topology but does not have to know details on each network domain resources to be allocated to that VCAN. This is a realistic assumption, given the autonomy of each network domain. That is why a negotiation is necessary between the initiator CAN Manager and other CAN Managers involved – to check if the local resources are sufficient. The monitoring at CAN layer and network layer is performed at an aggregated level.

Figure 2 shows an example of inter CAN Managers signaling (i.e., inter-controllers in SDN terminology).

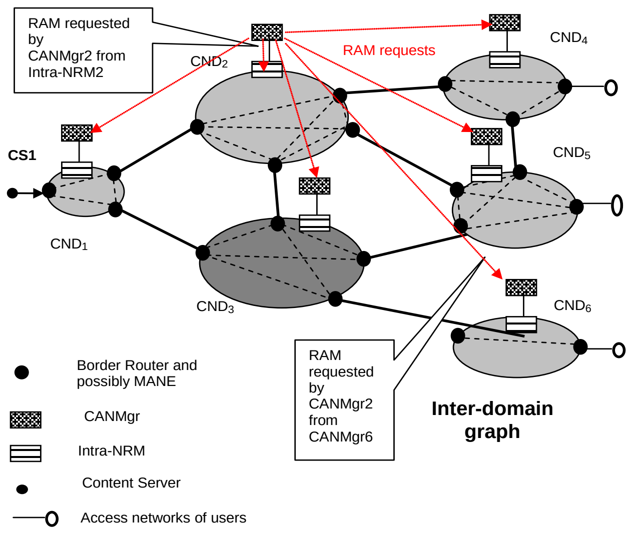


Figure 2. CAN Manager 2 issues random access memory (RAM) requests from each CAN Manager involved in a VCAN
Notations: CS- Content Server ; CND - Core Network Domain; CANMgr - Content Aware Network Manager; Intra-NRM – Intra-domain Network Resource Manager; RAM – (Domain) Resource Availability Matrix; MANE – Media Aware Network Element; CANMgr2- VCAN initiator

Note that in the initial VCAN request from the SP (to the initiator CAN Manager), the ingress and egress points of the VCAN are specified; additionally the initiator CAN Manager knows the inter-domain graph, so it can determine (in a first phase), which domains are involved in this VCAN. This is an internal algorithm running inside the initiator CAN Manager, whose details are out of this paper scope. The result of this is that the initiator CANMgr knows which domains are candidates to support this VCAN, so it knows to whom it should communicate and ask for resources information.

As an example, in Figure 2 the initiator CAN Manager 2, attached to Core Network Domain 2 asked (in hub style) the other involved CAN Managers (1, 3, 4, 5, 6) to deliver to it their network Resource Availability Matrices. Based on the received information, the initiator performs the VCAN mapping.

V. Simulation Model

The objective of this paper is the evaluation of the Management and Control signaling overhead, related to the negotiation activities between the actors: SP, CAN Managers, Intra-NRMs, when the number of network domains and CAN Managers is a variable parameter.

Given the complexity of the M&C subsystem, a simulation study has been developed. *Real Time Developer Studio* is a *Specification and Description Language* (SDL) simulator, developed by PRAGMADEV. It comes in two versions: SDL and SDL Real Time (SDL-RT) [19].

SDL-RT is based on ITU standard SDL (using *Extended Finite State Machine Model* - EFSM), extended with real time concepts. It is object oriented, has associated a graphical language and allows modeling real-time features. It combines static, dynamic and representations, supporting classical real time concepts, extended to distributed systems, based on standard languages. It retains the graphical abstraction brought by SDL while keeping the precision of traditional techniques in real-time and embedded software development. In SDL-RT, the C language is used to define and manipulate data. Therefore, it allows re-using legacy code written in C language.

The ALICANTE management simulation model consists in: *one Service Provider; N x CAN Managers, N x Intra_NRMs*, where the N variable is the number of network domains (e.g., 1 ...N_Max). Note that in practice, given the tiered structure of the Internet, the total number of domains (they might be autonomous systems) involved in an average length E2E communication is rather low.

The specific target is to evaluate the time spent from the instant when a Service Provider issues a VCAN request to an initiator CAN Manager, until the final confirmation of the VCAN installation is obtained by the Service Provider. The Service Provider can choose for its request any CANMgr as VCAN initiator, based on their proximity and involvement in the requested VCAN (or, other policy criteria).

The summary description of the management actions follows. The chosen CAN Manager, named afterwards Initiator CANMgr, will interrogate an inter-domain database containing information about inter-domain topology and network capabilities of the others domains, then run the *inter-domain mapping* algorithm. After this, it will communicate with each CANMgr identified by the inter-domain mapping algorithm as being involved in the requested VCAN, in order to find out its resources.

Note that the simulation model assumes parallelism in communication process from the initiator CAN Manager to the others (in "Hub" style). This is an important feature and design decision, assuring the scalability of the negotiation process.

Figure 3 describes the system processes model based on Extended Finite State Machine (EFSM) [20] represented in SDL-RT tool notation. It contains the global variables, the instance of each class and the other blocks involved.

The system SDL model consists of an *interDB* block, (used in simulation only, corresponding to an inter-domain database that contains inter-domain network topology), a SP_cloud block, associated with the SP/CP requestors, and ND (Number of Domains) CANMgr(s).

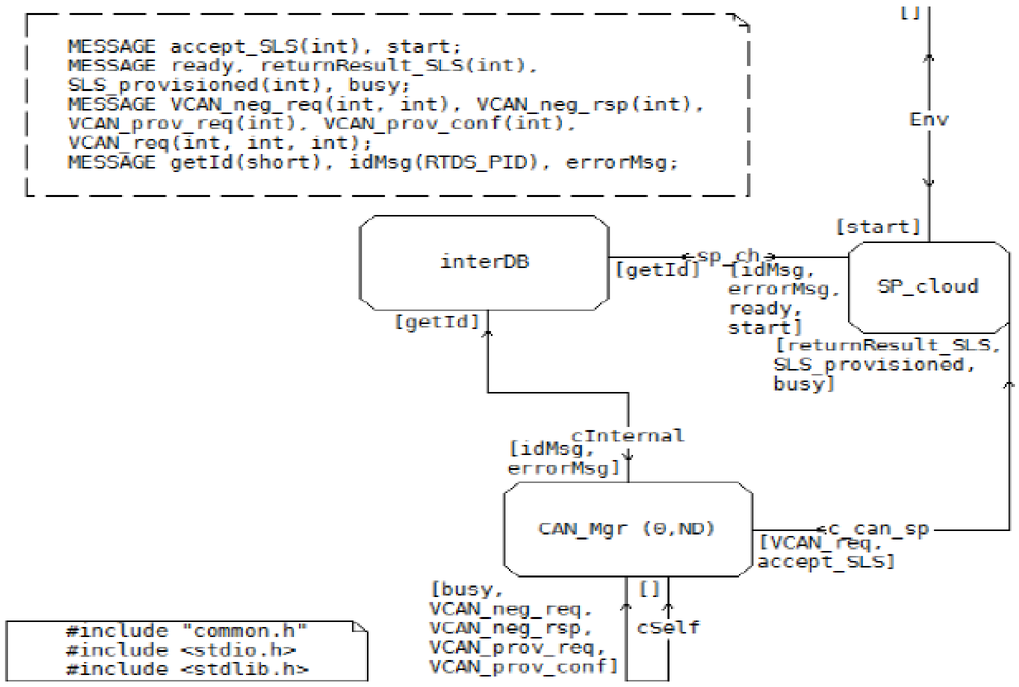
Figure 4 presents (just as an example sample) a part of the behavior of the CAN Manager EFSM. The typical SDL graphical notations can be seen, defining the interfaces and messages between blocks.

The works [11], [20] fully describe details of the Message *Sequence Charts* for the signaling process evolution between an initiator CAN Manager, and other CAN Managers involved in constructing a multiple domain VCAN. Figure 5 only presents, as an example, an actual sample of this signaling trace, extracted from the simulation tool results when running the system. It emphasizes a main phase of the VCAN construction: Service Level Specification negotiation.

The initiating CANMgr sends a *VCAN_neg_req* to each of the corresponding CANMgr and enter into *"negotiating"* state.

Each corresponding CANMgr checks its own capabilities (by running an intra-domain mapping algorithm), responds to initiating CANMgr with a *VCAN_neg_rsp* message, and transits to *"waiting_for_acceptance_ext"* state. The initiating CANMgr waits for all corresponding CAN_Mgrs to respond, then integrates the responses. Then it returns an integrated response (*return_result_SLS*) message to SP_cloud and waits for a decision. The above response message indicates to SP that all the requested resources are available and can be provisioned.

The Service Provider analyzes the response and sends a provision request to the initiating CANMgr, using the message *accept_SLS*. Then the initiating CANMgr sends a provision request message (*VCAN_prov_req*), to each of the corresponding CANMgr and waits for their confirmation response.



```
MESSAGE accept_SLS(int), start;
MESSAGE ready, returnResult_SLS(int),
SLS_provisioned(int), busy;
MESSAGE VCAN_neg_req(int, int), VCAN_neg_rsp(int),
VCAN_prov_req(int), VCAN_prov_conf(int),
VCAN_req(int, int, int);
MESSAGE getId(short), idMsg(RTDS_PID), errorMsg;
```

```
#include "common.h"
#include <stdio.h>
#include <stdlib.h>
```

[]
Env
[start]
interDB
sp_ch
[getId]
[idMsg, errorMsg, ready, start]
SP_cloud
[getId]
[returnResult_SLS, SLS_provisioned, busy]
cInternal
[idMsg, errorMsg]
CAN_Mgr (0,ND)
c_can_sp
[VCAN_req, accept_SLS]
[busy, VCAN_neg_req, VCAN_neg_rsp, VCAN_prov_req, VCAN_prov_conf]
[]
cSelf

Figure 3. The system model used in RTDS simulations

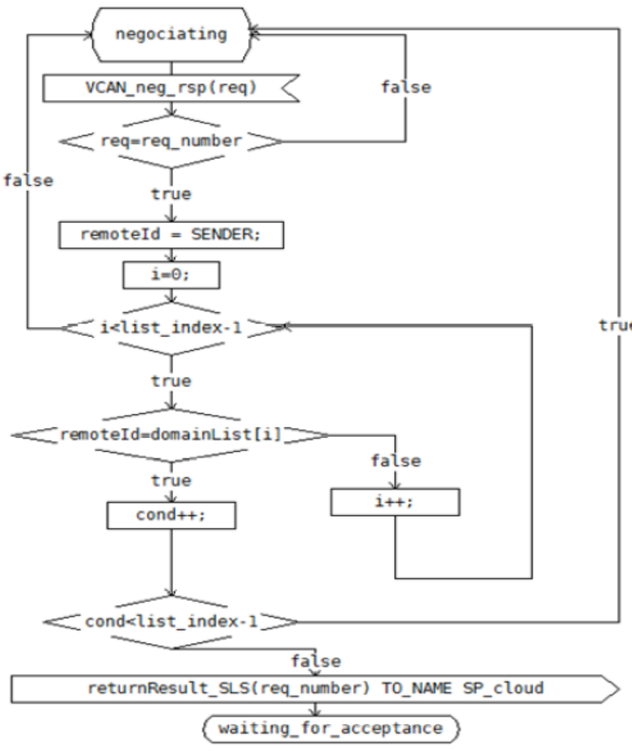


Figure 4. Sample example of the CAN Manager EFSM behavior

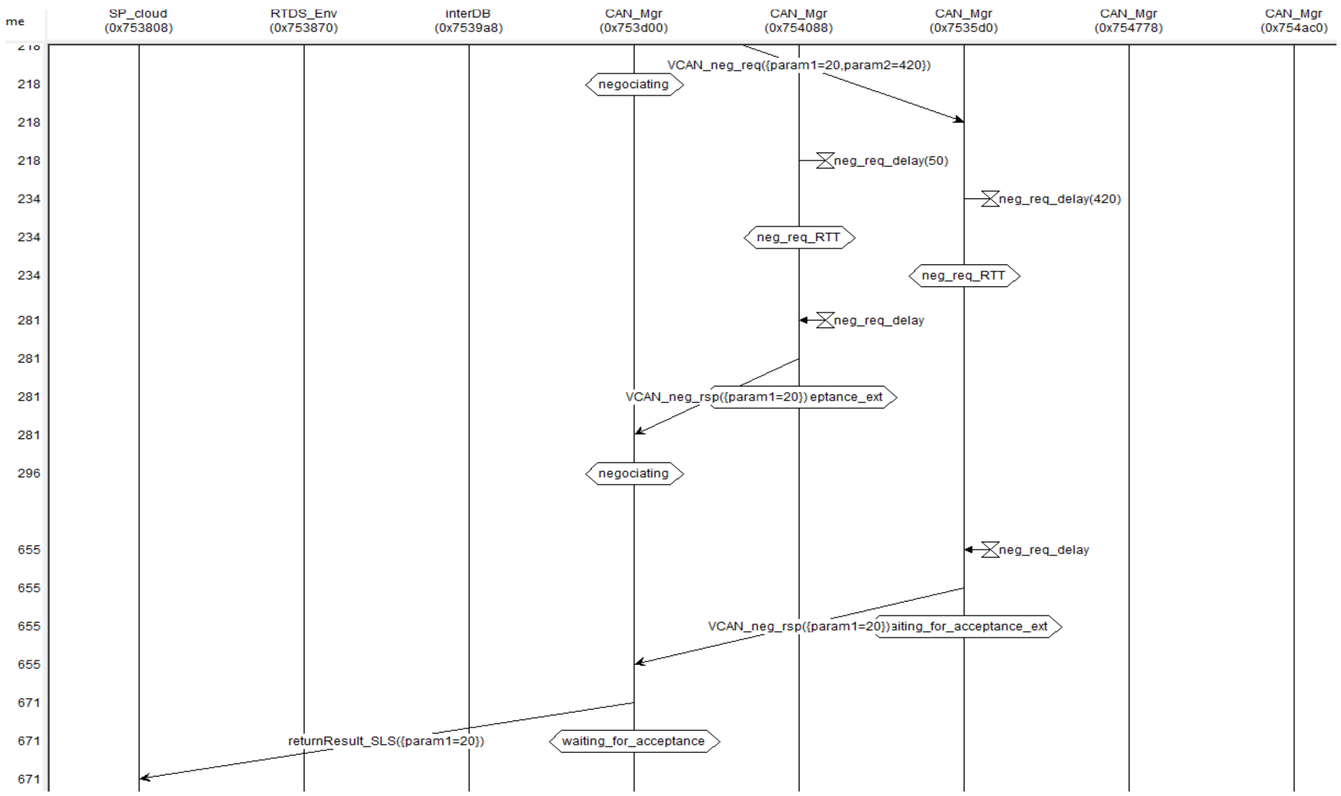


Figure 5. Signaling Message Sequence Chart -sample

VI. SIMULATION RESULTS

The simulations are focused on identifying the system behavior, and to determine a quantitative and qualitative estimation of the signaling time.

Being a real time simulator, the Real Time Developer Studio (RTDS) SDL-RT tool [18], uses the internal PC clock to estimate the time for each task/process from the system. Therefore, the results are defined in "ticks", which are relative time units.

The simulation model just simulates the time consumed by the inter-domain and intra-domain mapping algorithms, but it does not actually compute those algorithms (these computations are described elsewhere [17][18]). However, the result of the mapping algorithm, the chosen CAN Manager and the Round Trip Time (RTT) delay between two communicating CAN Managers are introduced in simulator using configuration files.

The simulation results presented here are extensions and can be considered as complementary to the ones presented in the initial paper [1].

While the simulation model uses an *abstract time clock*, in the experiments done, we can evaluate a *time unit* comparable to 1ms. However, in this study the trends and relative behavior instances (when different parameters are varying) are of more interest than the absolute values. The reason consists in the fact that a real implementation will involve different physical machines (belonging to the Network Provider) running the management software for CAN Managers; also the geographical placement of the domains have its importance w.r.t. quantitative results. WE are mainly interested in evaluating the scalability of the system. That is why we will use the time units (TU) when presenting the results.

Two sets of simulation runs have been performed:

a. considering a fixed delay (constant) RTT value for each corresponding CAN Manager involved

b. considering the same average value as the constant RTT, but with random jitter (uniform distribution). The case b. emulates a real situation where the CAN Managers are placed in different network domains and communicates via Internet. Table I presents some samples where the average delay for a CANMgr to respond to the initiator is 300 ms while a jitter of this time is +/- 100ms. The simulations have changed the number of domains involved, between 2 and 24, covering the use cases from small (w.r.t the number of domains involved) VCANs up to large ones.

The computing performance differences between the two machines (Personal Computers - PCs) are just qualitative criteria on evaluating the performance of a real CAN Manager machine when computing VCAN requests in ALICANTE environment.

Actually several runs have been performed with delay having values among: 100, 200, 300, 1000, 3000, 5000 ms.

This range of values has been selected so as to put into evidence two qualitative cases, concerning the impact upon the total signaling delay:

- processing time inside the various CAN Managers is dominant w.r.t. RTT;
- the RTT is dominant (e.g., 3000, 5000 ms) versus processing time.

Table I. Sample table showing different instants of terminating different partial actions, during a simulation with average delay (RTT) = 300 ms. The maximum number of domains has been 24.

Delay 300 TU (~ms)			Jitter +/-100 TU	N=24 domains	
# of Domains	Start time	VCAN_ req	VCAN_ neg_req	Return Result_ SLA	Stop time
2	344	390	484	921	1404
4	375	421	609	1045	1545
6	343	390	624	1123	1685
8	390	437	780	1388	2075
10	374	436	858	1482	2152
12	375	437	952	1716	2543
14	359	421	999	1810	2699
16	374	436	1107	2043	3104
18	375	437	1170	2200	3292
20	374	437	1279	2434	3635
22	375	437	1357	2621	3947
24	343	437	1420	2793	4212

In order to evaluate the processing power influence upon the total signaling time, the simulations are performed on two different machines, (i.e., named "Processor_1" - low power, and "Processor_2" - high power (Table II).

Table II. PC machine main hardware characteristics

PC configuration	Windows Experience Index	
	Processor_1	Processor_2
Processor	6	7.5
Memory(RAM)	5.9	7.8
Primary hard disk	5.9	7.9

Most of simulations are performed on a powerful machine, named "Processor_2". However, some of simulations are performed also on a slower machine, named "Procesor_1". These latter simulations serve as validation of results obtained from “Processor_2". As expected, the results from "Procesor_1" have bigger relative time values compared with the values from "Processor_2", given that the processing time is shorter on powerful "Processor_2" machine (see Table II).

A simplifying assumption, valid in all simulations, is that all messages issued by an entity arrives correctly at their destination (reliability of the communication has not been an objective of this specific study).

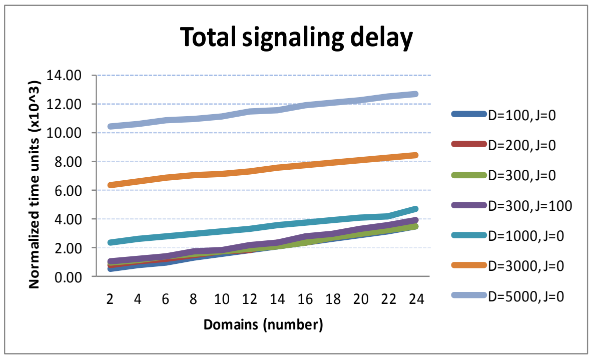


Figure 6. Total signaling delay, versus number of domains, for different values of the delay D (RTT); Processor_2.

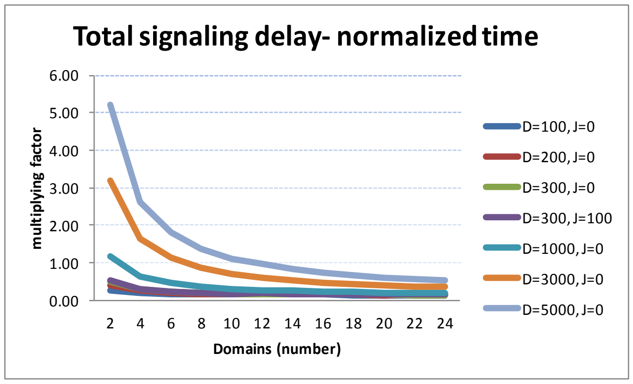


Figure 7. Normalized values (D/N): total signaling delay, Processor_2, versus number of domains, for different values of the delay D (RTT)

Figures 6 and 7 evaluate the performance of the overall system. They present the total signaling time (from the instant when the Service Provider issues a VCAN request to an initiating CAN Manager until the provisioning of the VCAN is finished and finally confirmed at SP). Figure 6 considers absolute values for delays while Figure 7 represents the behavior while using normalized values Different values for the delay (i.e., Round Trip Time – RTT) have been taken in the set of experiments.

Figure 7 shows that the normalized values of the signaling time are converging when the number of domains is increasing. If we consider absolute values of the total signaling time, one can see that the system can construct an average number of 10^2 – 10^3 VCANs/hour, which is quite sufficient in practice.

Three important conclusions can be extracted from the above results:

- The system is scalable w.r.t. number of domains involved, proved by the fact that all diagrams have approximately a linear behavior. This is the major result, showing that the management system can control large VCANs without significant signaling overhead;
- When the total transfer time (processing time plus propagation time through the internet) is dominant, the influence of the number of domains upon the total signaling time is lower (see the graphics for D = 300 ...5000).
- On the contrary, in cases of small average transfer time (e.g., D = 100, 200) the total signaling time increases linearly – with a higher slope when the number of domains increases.

Sets of results like the above may have utility for SP in establishing certain policies, when it makes the preliminary VCAN planning.

The following Figures (8 to 13) evaluate the relative time intervals consumed by different processing components of the overall signaling system. The objective is to identify those components, which produce more time consumption, in order to provide input data for system optimizations.

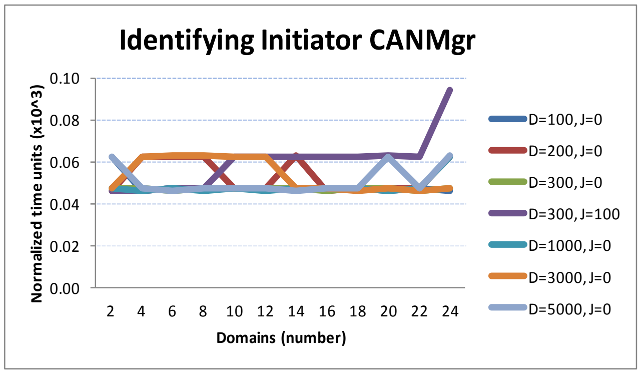


Figure 8. Processing time to identify the Initiator CAN Manager; Processor_2

Figure 8 presents the processing time consumed to identify the Initiator CAN Manager. These results validate the assumption that the number of domains do not influence this value (which is relatively low). The small variations shown in the figure are only statistical ones.

Figures 9 and 10 present the processing time consumed to determine the topology and identifiers of the CAN Managers involved in the required VCAN. One can see a linear increase of this time with the number of network domains; however the network communication transfer time is not involved, so the value is relatively low in the total budget of time.

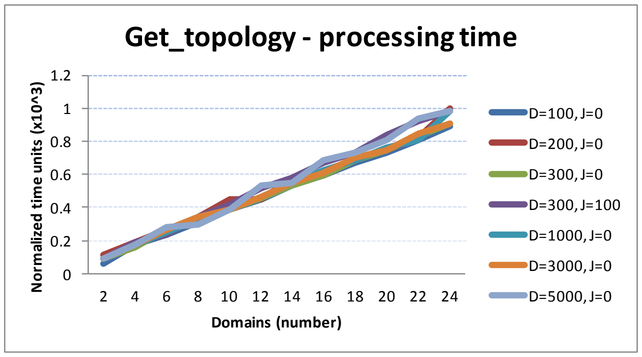


Figure 9. Processing time consumed to acquire the inter-domain topology; Processor_2

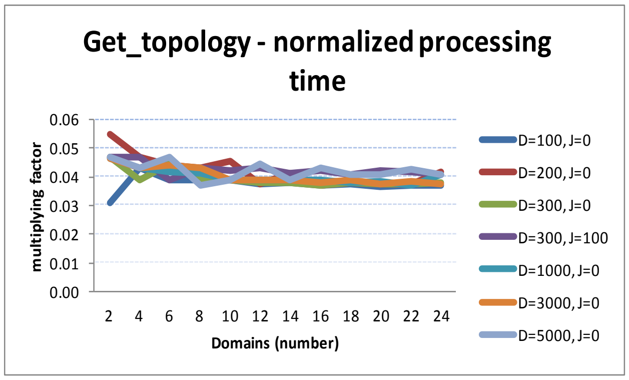


Figure 10. Normalized values: Processing time consumed to acquire the inter-domain topology; Processor_2

Figures 11 and 12 present the processing time consumed to perform SLS negotiations between the Initiator CAN Manager and its partners, i.e., other CAN Managers involved in the required VCAN. As expected, this set of actions has a major contribution to the total signaling time, because, additionally to the processing time, transfer through the network is involved. This is confirmed by the similarities between the diagrams of Figures 6-7 and those of the Figures 11-12.

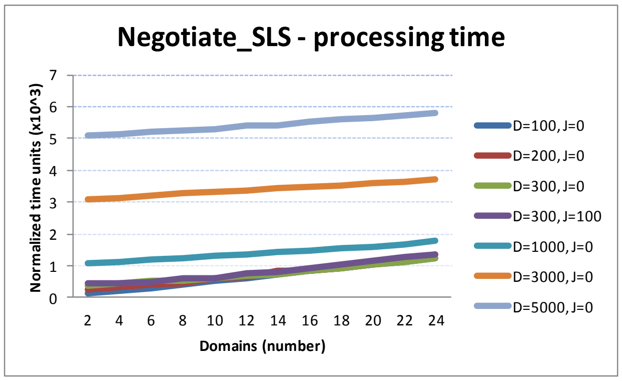


Figure 11. Processing time consumed to negotiate the Service Level Specification (SLS) contracts; Processor_2

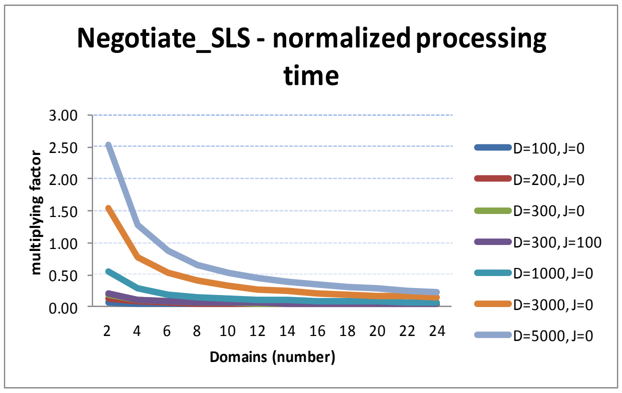


Figure 12. Normalized values: Processing time consumed to negotiate the Service Level Specification contracts (SLS); Processor_2

Note that additional variation for the values presented in Figures 11 and 12 could exist in practice, depending on how frequently a given CAN Manager is updating its Resource Availability Matrix (result of the dialogue CAN Manager and its associated Intra-domain Network Resource Manager). Here, one may have independent policies like push/pull and synchronous or asynchronous communication type.

Figures 13 and 14 present the processing time consumed to provision (install all required configurations in the MANE and core routers) the VCANs in the network. This set of actions has a major contribution to the total signaling time, because the time is consumed by each CAN Manager to trigger installation of the VCAN configurations in the routers (via Intra_NRM) and also the transfer through the inter-domain network is involved for the signaling messages. This is confirmed by the similarities between these diagrams and those of the Figures 11-12. Optimizations in this area can contribute significantly to the overall system performance.

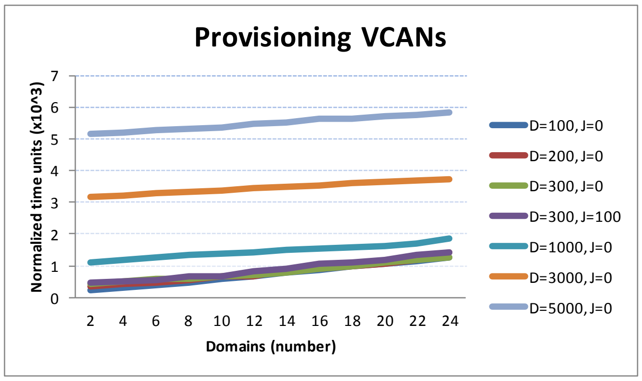


Figure 13. Processing time consumed to provision VCANs in the network; Processor_2

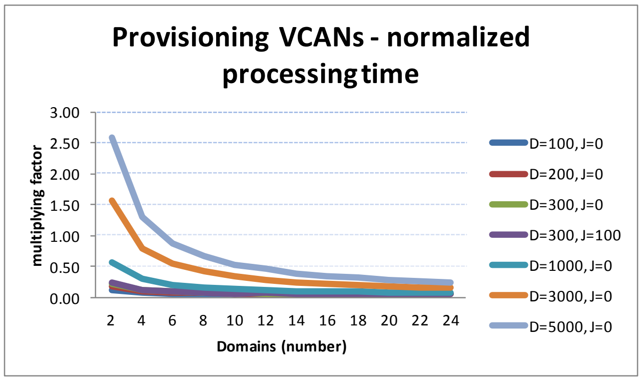


Figure 14. Normalized values: Processing time consumed to provision VCANs in the network; Processor_2

Figure 15 shows a comparison between simulations on "Processor_1" and "Processor_2". The behavior of the systems is similar (but having different slopes in the diagram), while in the case of a slower Processor_1, only the convergence value is different (4500 for "Processor_1" and 2500 for "Processor_2"). Again, the convergence is present, and the difference on convergence value is the result of different computing time inside CANMgr.

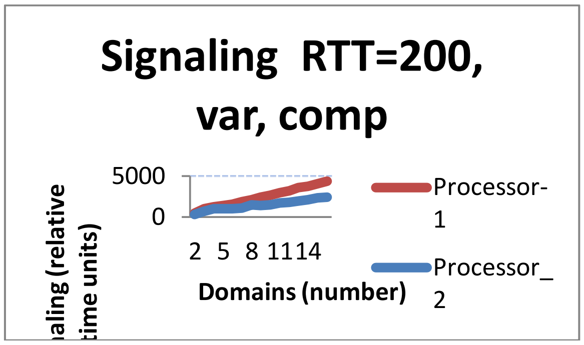


Figure 15. Comparison of the total signaling times; average D=RTT=200; Processor_1 versus Processor_2.

In order to have an additional validation from statistical point of view, a set of simulations was performed on "Processor_2", using different seeds. The simulation results are shown in Figure 16. The same overall behavior is obtained; the small difference of the convergence value occurs due to the seed influence on simulator internal algorithm, shown on the relative time units obtained on each simulation.

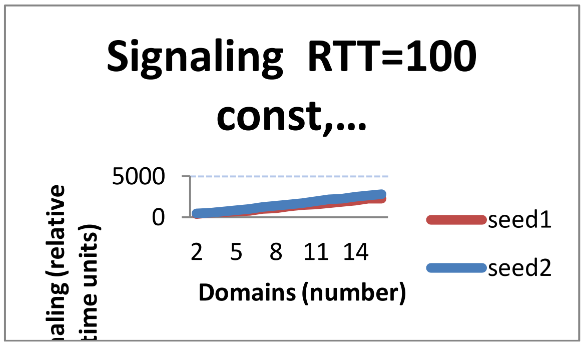


Figure 16. Total signaling time for RTT=100 constant, different seed, Processor_2

For scalability extended evaluation purpose only, a set of simulations was performed, considering 500 domains (CANMgr). As shown in Figure 17 the linear behavior is also exposed up to a high number of domains (N=500).

Other simulations have shown that the components of the total signaling time have roughly the same relative weights for large number of domains experiments, similar to those presented in Figures 9 – 14.

The overall set of simulations (including the extended range ones) provides a confirmation of the initial assumptions about the signaling system performing the multi-domain VCAN construction: the system is scalable w.r.t. number of domains involved. The proof is provided by the fact that all diagrams have approximately a linear behavior, so the management system can control VCANs spanning multiple domains with low signaling overhead.

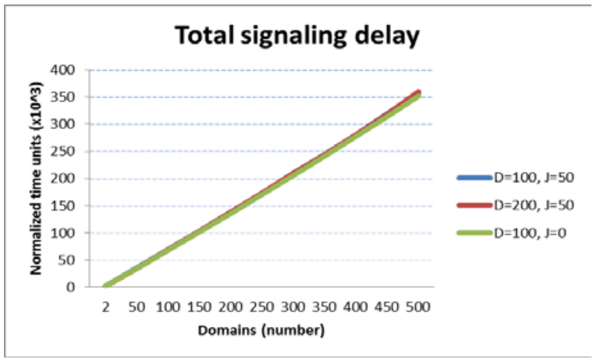


Figure 17. Total signaling delay, versus number of domains, for different values of the delay D (RTT) ; Processor_2, N=500 domains

The major contribution to the total time spent for a VCAN construction is brought by the VCAN provisioning phase, given that three components intervene: a. transfer time (in both directions) through the inter-domain networks, needed for communication between the Initiator CAN Manager and the other CAN Managers involved; b. processing/computation time to determine the resources to be provisioned in each CAN Manager involved in a multiple-domain VCAN; c. transfer time needed by each Intra-NRM to inject the VCAN configurations in all its MANE and core routers. In practice, optimization techniques can reduce the time spent by these components.

VII. CONCLUSIONS

This paper extended the results presented initially in ICNS 2014 paper [1], providing a simulation model and results concerning the scalability of the multi-controller communication subsystem as a functional management component of a media delivery ecosystem. It is shown that the proposed control approach is conveniently feasible in a multi-domain network environment.

First, the management architecture of a multi-domain media delivery system (in particular ALICANTE [10][11]) system has been outlined, showing its partial similarity with the SDN multi-controller architecture.

The architectural equivalence has been analyzed between an SDN regional controller and the pair {CAN Manager and Intra-domain Network Resource Manager} considered in this study. Horizontal scalability problems appear in both SDN and ALICANTE multi-controller environments.

A simulation model based on Extended Finite State Machines approach has been constructed for the ALICANTE management architecture, aiming to evaluate mainly the total signaling time for Virtual Content Aware Networks negotiation and provisioning over a multi-domain environment. Extensive sets of simulation runs have been performed for various network domains, network transfer conditions and processing times.

The main finding of this article is that signaling system is horizontally scalable, versus the number of network domains involved. This has been proved by the approximate linear behavior of the total signaling time. The components of the total signaling time have been also identified. Quantitative metrics have been determined in different use cases, emphasizing the dominance of the processing time or the network transfer times.

Further work should evaluate the capacity of the ALICANTE architecture to get closer to the SDN approach, and also methods to integrate the VCAN construction particular problem into more general SDN controller framework. Another direction is to investigate how one controller can command a given number of network elements (routers) by using a vertical protocol (similar to OpenFlow).

ACKNOWLEDGMENT

This work has been partially supported by the FP7 European research project ALICANTE (FP7-ICT-248652) and partially by the projects POSDRU/107/1.5/S/76909.

REFERENCES

[1] E. Borcoci, M. Constantinescu, and M. Vochin, “Multi-controller Scalability in Multi-domain Content Aware Networks Management,” ICNS 2014 Conference, http://www.iaria.org/conferences2014/ProgramICNS14.html, pp. 62-68.

[2] J. Schönwälder, M. Fouquet, G. D. Rodosek, and I. Hochstatter, “Future Internet = Content + Services + Management,” IEEE Communications Magazine, vol. 47, no. 7, pp. 27-33, July 2009.

[3] T. Anderson, L. Peterson, S. Shenker, and J. Turner, “Overcoming the Internet Impasse through Virtualization,” Computer, vol. 38, no. 4, pp. 34–41, Apr. 2005.

[4] J. Choi, J. Han, E. Cho, T. Kwon, and Y. Choi, “A Survey on Content-Oriented Networking for Efficient Content Delivery,” IEEE Communications Magazine, March 2011.

[5] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking Named Content,” CoNEXT ’09, New York, NY, pp. 1–12, 2009.

[6] N. McKeown et. al., “OpenFlow: Enabling Innovation în Campus Networks,” March 2008, -http://www.openflow.org/documents/openflow-wp-latest.pdf.

[7] B. A. Nunes, M. Mendonca, X. N. Nguyen, K. Obraczka, and Thierry Turletti, “A Survey of Software-Defined Networking: Past, Present, and Future of Programmable Networks,” January 2014, http://hal.inria.fr/hal-00825087

[8] OpenFlow Switch Specification, V 1.3.0 (Wire Protocol 0x04), June 25, 2012, https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-spec-v1.3.0.pdf

[9] S. H. Yeganeh, A. Tootoonchian, and Y. Ganjali, “On Scalability of Software-Defined Networking”, IEEE Communications Magazine, pp. 136-141, February 2013.

[10] FP7 ICT project, “MediA Ecosystem Deployment Through Ubiquitous Content-Aware Network Environments,” ALICANTE, No248652, “D2.1: ALICANTE Overall System and Components Definition and Specifications”, http://www.ict-alicante.eu/.

[11] C. Cernat, E. Borcoci, and V. Poenaru, “SLA Framework Development for Content Aware Networks Resource Provisioning,” AICT 2013 Conference, http://www.thinkmind.org/index.php?view=article&articleid=aict_2013_8_30_10084

[12] K. T. Cheng and A. S. Krishnakumar, "Automatic Functional Test Generation Using The Extended Finite State Machine Model," International Design Automation Conference (DAC), ACM, 1993, pp. 86–91.

[13] A. Pathan and R Buyya, “A Taxonomy and Survey of Content Delivery Networks,” http://cloudbus.org/reports/CDN-Taxonomy.pdf. 2008.

[14] B. Raghavan, T. Koponen, A. Ghodsi, M. Casado, S. Ratnasamy, and S. Shenker, “Software-Defined Internet Architecture: Decoupling Architecture from Infrastructure,” HotNets-XI Proceedings of the 11th ACM Workshop on Hot Topics in Networks, 2012, pp. 43-48, doi: 10.1145/2390231.2390239.

[15] A. Tavakkoli, M. Casado, and S. Shenker, “Applying NOX to the Datacenter,” Proc. ACM HotNets-VIII Wksp., 2009.

[16] H. Yin, H. Xie, T. Tsou, D. Lopez, P. Aranda, and R. Sidi, draft-yin-sdn-sdni-00.txt, ”SDNi: A Message Exchange Protocol for Software Defined Networks (SDNS) across Multiple Domains,” June 2012.

[17] E. Borcoci, R. Miruta, and S. Obreja, “Multi-domain Virtual Content-Aware Networks Mapping on Network Resources,” EUSIPCO Conference, 27-31 August 2012, Bucharest, http://www.eusipco2012.org/home.php

[18] R. Iorga, E. Borcoci, A. Pinto et. al., “Management driven hybrid multicast framework for content aware networks,” IEEE Communications Magazine, Vol. 52, January 2014, pp. 158-165.

[19] http://www.sdl-rt.org/, 12.11.2014

[20] ALICANTE Deliverable D8.3, “Trials and Validation,” http://www.ict-alicante.eu/validation/delivrables/, 14.10.2014

Performance of Meshed Tree Protocols for Loop Avoidance in Switched Networks

Kuhu Sharma, Bill Stackpole, Daryl Johnson, Nirmala Shenoy and Bruce Hartpence
College of Computing and Information Sciences
Rochester Institute of Technology,
Rochester, NY, USA
kxs3104@rit.edu, Bill.Stackpole@rit.edu, daryl.johnson@rit.edu, nxsvks@rit.edu, bhhics@rit.edu

Abstract—**Loop free frame forwarding in layer 2 switched networks that use meshed topologies to provision for link and path redundancy is a continuing challenge. The challenge is addressed through special protocols at layer 2 that build logical trees over the physically meshed topologies, along which frames can be forwarded. The first such protocol was based on the spanning tree. The spanning tree protocol (STP) had high convergence times subsequent to topology changes. Rapid STP and IETF RFC 5556 *Transparent Interconnection of Lots of Links* (TRILL) on Router Bridges (RBridges) were then developed to reduce the convergence times. RSTP continued to use the spanning tree while TRILL adopted link state routing to support a tree from every switch. TRILL introduces high processing complexity into layer 2 networks. In this article a new meshed tree algorithm (MTA) and a loop avoidance protocol based on the MTA, namely the meshed tree protocol (MTP) are discussed. The MTA allows constructing several overlapping trees from a single root switch. This speeds up convergence to link failures. The MTP proposes a simple numbering scheme to implement meshed trees – thus, the processing complexity is low. The specification for the MTP is currently an ongoing IEEE standard Project 1910.1. In this article the operational details of MTP are presented and its performance evaluated and compared with RSTP.**

Keywords- Loop Avoidance, Switched Networks, Meshed Trees Protocol, Link Failure and Recovery

I. INTRODUCTION

Loop free forwarding is a continuing challenge in layer 2 switched networks. The need for link and path redundancy to provide a continuous communications path between pairs of end switches in the event of switch or link failure requires a physical network topology that is meshed. However, the physical loops in a mesh topology cause broadcast storms when forwarding broadcast frames. Hence, it is important to have a logical tree topology overlaid on the physical meshed topology to forward broadcast frames. The first such logical loop-free forwarding solution was based on the Spanning Tree. Radia Perlman [1] proposed the specifications of a protocol called the Spanning Tree protocol (STP) based on the Spanning Tree Algorithm (STA). A spanning tree in a switched network was constructed by logically blocking some of the switch's ports from forwarding frames. The basic STP had high convergence times during topology changes. Rapid Spanning Tree Protocol (RSTP) was developed to reduce the convergence times in the basic STP. However, RSTP still retained some of the inefficiencies of spanning trees, one of which is forwarding all frames through the root and a second is the root re-election on topology changes. Radia Perlman then proposed Transparent Interconnection of Lots of Links (TRILL) on RBridges (router bridges) to overcome the disadvantages of STA-based loop avoidance. This came at the cost of processing overhead and implementation complexity as TRILL used the Intermediate System to Intermediate System (IS-IS) routing protocol at layer 2. The goal was to use optimal paths for frame forwarding between pairs of switches and also avoid root election. IS-IS is a link state routing protocol that can operate independently of the network layer. The TRILL protocol was implemented above layer 2 and used special headers to encapsulate the TRILL and IS-IS related link state routing messages. The bridges were called RBridges as they implemented routing protocols. TRILL on RBridges is currently an Internet Engineering Task Force (IETF) draft [2]. Shortest Path Bridging (SPB) was developed along similar lines by IEEE 802.1aq, also adopting the IS-IS routing protocol at layer 2. TRILL is considered as superior to RSTP due to the redundant links that are established. Shortest Path Bridging (SPB) based loop avoidance primarily targets high switching speeds as required in service provider and backbone provider networks. Thus, two versions of SPB have been specified, SPBV (SPB with VLAN Ids) and SPBM (SPB with MAC addresses) [3] for service provider networks and backbone provider networks, respectively.

The premise for the loop avoidance solutions discussed above is that a single logical tree from a root switch that operationally eliminates physical loops is necessary to resolve the conflicting requirements of physical link redundancy and loop free frame forwarding. Under this approach, in the event of link failure, the tree has to be recomputed. While spanning tree is a single tree constructed from a single elected root switch, the Dijkstra algorithm used in IS-IS based routing builds a tree from every switch. The operation of IS-IS requires link state information in the entire network to be disseminated to every switch so that each switch can compute its own tree by running the Dijkstra algorithm on the connectivity information that it collects and stores in a Link State database. On topology changes, link state information must again be disseminated to all switches and the Link State database should be stable for some time before the Dijsktra algorithm can be run. During this period the frame forwarding information is unstable. TRILL on RBridges uses a hop count to avoid looping of frames.

In [4], a novel meshed tree algorithm (MTA) and the associated Meshed Tree Protocol (MTP) was introduced. Its performance was evaluated and compared with RSTP. Unlike the trees discussed above the MTA allows construction of multiple trees from a single root by using the multiple paths provisioned by the meshed topology. Loop-free frame forwarding can happen using any one of the multiple trees. The MTP based on the MTA allows for creation and maintenance of *multiple* overlapping tree branches from *one* root switch. The multiple branches mesh at the switches, and thus on the failure of a link (or branch) the switch can fall back on another branch without waiting for re-computation of the tree. Frame forwarding can continue while the broken branch is pruned. This eliminates temporary inconsistent topologies and latencies resulting from tree reconstruction. The premise of the MTP is to leverage the multiplicity of connections in a meshed topology by constructing and maintaining several trees from a single root concurrently [5-9]. Thus, the MTA addresses the convergence issues facing STA based protocols and also avoids the complexity of IS-IS based loop avoidance solutions. In addition, there can be multiple root switches, where each root supports its own meshed trees. This extends the MTA to Multi Meshed Trees (MMT), which can be used to introduce redundancy in the event of root failure. This feature of MTP is not covered in this article.

The novel feature of the MTA is implemented through a simple numbering scheme. Meshed Tree Virtual IDs (MT_VIDs) are allocated to each switch in the network. The MT_VID acquired by a switch defines a tree branch or logical frame-forwarding path from the root switch to that switch. A switch can acquire multiple MT_VIDs based on the MT_VIDs advertised by its neighboring switches and thus join multiple tree branches providing a switch with several paths to the root switch. In this way, meshed trees leverage the redundancy in meshed topologies to set up several loop-free logical frame-forwarding paths. No ports are blocked from forwarding frames.

In this paper, some basic operational specifications of the MTP are presented. These include meshed tree creation through the use of MT_VIDs, the limits on the level of meshing, the criteria and process for a switch to forward broadcast and unicast frames, and the handling of link failures. The specification of the MTP in this article is limited to customer VLANs where RSTP is the primary candidate solution. Thus, the performance of the MTP is evaluated and compared with RSTP. The comparison was conducted using OPNET simulation tool [10]. Though TRILL is considered as another candidate protocol for RSTP replacement, models of TRILL were not available for a comparative study. However, under Section II.D a detailed operational comparison of TRILL with the MTP is provided.

The significant improvement in the convergence times and the hops taken by frames to reach destinations indicate the superior features of the MTP. The operational simplicity of the MTP also provides advantages over complex Link State solutions. MT loop free forwarding at layer 2 is currently an IEEE project (1910.1) under the IEEE 1910 working group [11] lead by the authors. The rest of the paper is organized as follows. Section II discusses related work in the context of STP and *Link State* based solutions highlighting the comparable features of MT based solutions. In Section III, operational details of the MTP are presented. Section IV describes the optimized unicast frame forwarding schema adopted in the MTP. Section V provides the link failure handling mechanism adopted in the MTP. Section VI provides the simulation details and performance results. Section VII follows with conclusions.

II. RELATED WORK

In this section, we discuss the two primary techniques proposed for loop resolution in layer 2 switched networks. The first of these is based on the Spanning Tree Protocols (STP and RSTP) and the second is based on Link State (LS) Routing namely the TRILL on RBridges. This article does not describe all the operational details as such information is publicly available [12-14].

A. Protocols Based on Spanning Tree Algorithm

Both the STP and RSTP are based on the STA. To avoid loops in the network while maintaining access to all the network segments, the bridges compute a spanning tree after collectively electing a root bridge. For root election bridgeIDs are used. In (R)STP, each bridge first assumes that it is the root and announces its bridgeID. Upon receiving the bridgeID, neighbors compare it with their bridgeID and allow the bridge with a lower bridgeID to continue as a root. The unique bridgeID is a combination of a bridge priority and the bridge's medium access control (MAC) address. A bridge may supplant the current root if its bridgeID is lower.

One major disadvantage of STA based protocols is that all traffic flows via the root switch. It is thus important to have a root switch that has adequate processing capability and an optimal location within the topology. For this purpose the priority field in the bridgeID can be manually set by an administrator. Once a root bridge is elected, other bridges then resolve their connection to the root bridge by listening to messages from their neighbors. These messages include the path cost information from the root bridge. Bridges accept a connection to another bridge based on the lowest path cost. With the STP, other ports are blocked from frame transmission. Within a network deploying RSTP these ports are maintained in readiness (alternate, backup) to takeover on the failure of the unblocked ports in RSTP.

The STP has high convergence times after a topology change. To reduce the convergence times the *Rapid Spanning Tree* protocol (RSTP) was proposed [12]. The RSTP is a refinement of the STP and therefore shares most of its basic operation characteristics, with some notable differences including: 1) the detection of root bridge failure is done in 3 'hello' times, 2) response to Bridge Protocol Data Units (BPDUs) are sent only from the direction of the root bridge, allowing RSTP bridges to 'propose' their spanning tree information on their designated ports. The second feature allows the receiving RSTP bridge to determine if the root information is superior, and set all other ports to 'discarding' and send an 'agreement' to the first bridge. The first bridge can rapidly transition that port to forwarding and bypass the traditional listening/learning states. 3) Lastly, backup details regarding the discarding status of ports are maintained to avoid failure timeouts of forwarding ports.

STP and RSTP: STA based implementation is simple as the spanning tree is executed with the exchange of BPDUs among neighboring bridges that carry *tree formation* information. Several disadvantages of STA based protocols are noted by the inventors of STA [14]. These include: 1) Traffic concentration on the spanning tree path, as all traffic follows the tree even when other more direct paths are available. This causes traffic to take potentially sub-optimal paths, resulting in inefficient use of the links and reduction in aggregate bandwidth. 2) Spanning tree is dependent on the way the bridges are interconnected. Small changes due to link failure can cause large changes in the logical spanning tree topology. Changes in the spanning tree take time to propagate and converge, especially for non-RSTP protocols. 3) Though IEEE 802.1Q describes multiple spanning trees, this requires additional configuration, the number of trees is limited, and the defects previously noted apply within each tree [3].

B. TRILL Protocol on RBridges

The TRILL protocol overcomes many of the shortcomings in STA based protocols. Convergence times are improved by supporting a tree from each switch. TRILL incorporates the routing functionality of layer 3 by using the IS-IS protocol [13, 14] at layer 2. The IS-IS protocol is used to compute pair-wise optimal paths between two bridges. The computed pair-wise optimal paths is used for forwarding frames at layer 2. Thus, the frame forwarding inefficiency in STA based protocols is avoided. Inconsistencies and loop formations during topology change can occur but are overcome by a *hop count* used in inter-bridge forwarding. TRILL encapsulates link state routing messages of IS-IS in special headers and uses special protocols to learn end station addresses.

Advantages and Disadvantages of TRILL: Advantages of the TRILL protocol include: 1) Frames are forwarded via an optimal path. 2) Transit frames are routed with a hop count, thus temporary loops will result in frames being discarded when the hop count reaches zero. 3) Route changes can be made quickly and safely based on local information. The disadvantages of IS-IS based protocols are: 1) They have to encapsulate all messages required for the operation of IS-IS. 2) The operations of IS-IS are distinct from layer 2 operations and VLANs in layer 2. This adds to the processing complexity at layer 2 which is compounded by the need for integrated operations of layer 2 and IS_IS routing functions. 3) All link state routing protocols require that Dijkstra algorithm be run only after the Link State database has stabilized for a certain time interval after the last link state update received. During this time the forwarding (routing) operations are unstable and this contributes to the convergence time of the network topology at layer 2. During this time, looping packets cannot be avoided which required IS-IS based solutions to include a hop-count to discard such packets. Dijsktra algorithm is also known for its processing complexity [15], which is proportional to the number of switches / links.

C. The Meshed Tree Protocol

Single-tree like structures imposed on topologies reduce or eliminate loops but also create an environment in which there are failover delays to alternate links. These topologies also lack redundancy or the ability to load balance. Protocols such as SPB and TRILL build trees from all nodes to alleviate these problems. However, as redundancy is introduced the complexity becomes very high due to the creation and maintenance of as many trees as there are switches. The MTP seeks to address these same issues with less complexity and even shorter failover times upon discovery of link failure. The core of the protocol is the ability of each switch to be a member of more than one tree. This provides path redundancy and quick fail-over to the redundant paths on link failure detection. Ports are not blocked which allows for optimized frame forwarding paths. Root redundancy requirements in single meshed-tree based on the MTP can be addressed by multiple meshed-trees (MMT) [5 - 9], where several switches can be roots and each can support a meshed tree. The number of roots can be optimized to improve redundancy and performance while keeping the complexity low.

D. Comparison with Link State Protocols

In the case of TRILL on RBridges optimal pairwise paths are computed and used for frame forwarding. However, the processing complexity has increased by several orders of magnitude. In the case of single meshed tree MTP, optimal paths can be computed based on the MT_VIDs acquired by the switches. Since switches may not record all MT_VIDs offered, some paths may not be the shortest.

In terms of convergence, link state routing requires all link state information to be flooded to all switches. Subsequently the Dijkstra algorithm will be run to compute the forwarding paths. During this time the source address table (SAT) may not be updated and could result in unstable operation. Using the MTP, the tree is built using information received from neighbor switches and flooding of information is avoided for tree resolution. In the event that tree pruning is required, the switches can still use the backup paths to forward frames.

Table I lists the major difference between TRILL and MTP.

Table I. Comparison of 'MTP on Bridges' vs 'TRILL on RBridges'

Feature	TRILL on RBridges	Meshed tree on bridges
Tree structure	• One shortest path spanning tree originating at the root Rbridge • Each Rbridge is present on only one branch of a single tree originating from a root bridge	• Several overlapped spanning trees with one of them being the shortest path spanning tree • Each bridge can reside on multiple branches of a single meshed tree originating from a root bridge
Multiple trees originating at different bridges	Possible	Possible
Knowledge of network topology	required	NOT required
Flooding of topology messages	required	NOT required
Action on link failure and addition /removal of bridges and links	• Generate link state updates and disseminate. • Flood topology control messages	• Repair locally. • Inform bridges downstream that have an MT_VID which is derived from the lost MT_VID. • Build tree branches as nodes join
Formation of temporary loops	Yes. Loop is broken when hop count (6 bits in the header) reaches 0.	Loop formation prevented
Avoidance of loop formation	Not completely avoided. Uses hop counts	Avoided due to the numbering scheme
Unicast frames (known destination address)	• Forwarded on pair-wise optimal paths determined by the link state routing protocol if End System Address Distribution Information (ESADI) is used. • Next hop path should be specified. • Encapsulated in TRILL header • Every Rbridge that forwards decapsulates and encapsulates again	• Neighboring bridges can forward directly to the appropriate port. • Forwarded on the optimal path decided by primary VID tree at the originating bridge.
Multicast traffic Unicast frames (destination unknown)	• Forwarded on distribution trees, using multi pathing to multiple destination. • Tree pruning advised (no specifications provided)	• Can follow the current process using multicast addresses at layer 2. • Meshed tree at originating bridge can be used.
End node address learning	• Open the internal Ethernet frame to determine the source address • Use ESADI protocol and inform all RBRridges	• Learn from source address as no encapsulation is used • Can exchange infromation between neighboring switches.
Computing complexity of Dijkstra's algorithm	• O(n^2) in a dense network for node selection with 'n' nodes. • O(m) for edge (link) updates with 'm' edges • O(m log n) using an adjacency list representation and a partially ordered tree data structure for organizing the set of edges [15].	O(1) –See Appendix A
Implementations	• Dynamic nickname protocol to reduce TRILL header • Topology control message dissemination • Encapsulation and de-encapsulation at forwarding Rbrdiges. Every transit frame has to be encapsulated with an external Ethernet header. Overhead per encapsulation equals 144 bits • End Station Address Dissemination Information (ESADI) protocol is optional • Election of a designated Rbridge per link • Designated VLAN required for Rbridge communication • Differentiate between IS_IS at layer 2 and layer 3 • Requires 'reverse path forwarding check" to control looping traffic **See schematic in Appendix B**	• Replace the ST algorithm with the MT algorithm. • Define software to run the MT algorithm • Works on the same principle as STA. MT_VIDs will be sent in BPDUs. **See schematic in Appendix B**

III. THE MESHED TREE PROTOCOL

The *MTA* allows construction of logically *meshed trees* from a single root switch in distributed manner using local information shared among neighbor switches [5-9]. In this article, MTP operations related to the construction of meshed trees are described. The discussion presented in this article does not include the election of a root bridge as the focus is on the loop resolution / avoidance feature of MTA based protocols. Hence we assume a designated root bridge.

Bridge ID: For the operation of the MTP bridgeIDs are necessary. These have to be unique only within the switched network. The MT_VIDs of switches are derived by appending the outgoing port number to the MT_VID of the switch that offers an MT_VID to a downstream switch. The root switch uses its unique ID as its MT_VID, thus the first value in the MT_VID acquired by other bridges will be the root bridgeID. In this article without loss of generality we used a single digit ID for the root switch though a simple MAC address derivative could be used.

Because of the way in which MT_VIDs are constructed, an MT_VID describes a path that connects a bridge to the root bridge. In a single physical meshed topology, a switch can be associated with more than one MT_VID and thus:

- A *Meshed Tree* could contain ***all*** of the possible paths from the root switch to each switch in the topology.
- More than one path to each switch can coexist

Consider a three-switch single loop topology shown in Fig. 1. In the upper left is the physical loop topology. In order to prevent traffic from looping, we might impose any one of several logical tree topologies like those shown. In the upper right, the topology is optimized for transmissions associated with switches connected to the root. But in the lower left and lower right, the topology is optimized for nodes connected to switches A and B, respectively. By themselves, these three logical topologies do not provide for redundancy. The MTA allows for building and using all of the logical trees simultaneously and because multiple pathways are pre-established, *failover times to redundant links are near zero.*

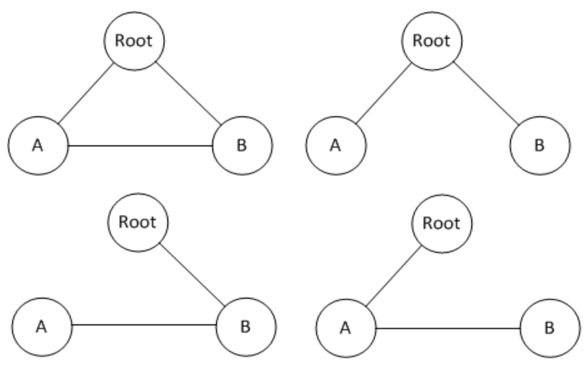


Figure 1. One physical meshed topology - three logical tree topologies

A. Basic Protocol Operation

The topology resolved using the MTP will support overlapping trees that are created and maintained through the MT_VIDs. A Meshed Tree Switch (MTS) that has membership on a tree will be assigned at least one MT_VID that is associated with that tree and a particular path back to the root. Significantly, switches having more than one pathway back to the root will have primary, secondary, tertiary, etc., memberships in multiple trees, each having a separate and unrelated MT_VID. MT_VIDs are stored in a table and have an association with ports through which they were established. Examples of trees from a single root and associated MT_VIDs are shown in Fig. 2.

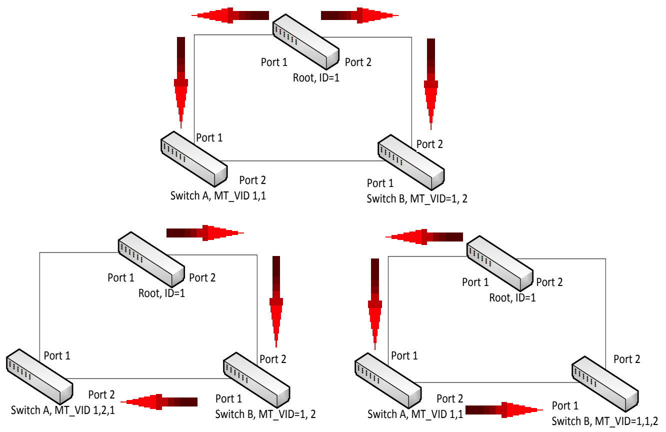


Figure 2. MT topologies and MT_VID Creation

On the top of Fig. 2 it can be seen that the topology is optimized to the root. The MT_VIDs and the tree are derived based on this perspective. However, in a looped topology, the downstream or child switches have alternate paths. In the bottom left and bottom right, switches A and B also have MT_VIDs that would be derived in these alternate logical tree topologies. Another way to look at this is to consider the traffic that might flow between switches A and B. Clearly, the topology that would be derived per spanning tree would be suboptimal as all traffic must first flow to the root switch and then back down. It is noteworthy that these alternate paths might be used to optimize transmissions between the hosts connected to the switches. Another important aspect of the MTP is that MTS's do not populate the SAT in the traditional manner; learning the source addresses of end hosts based on the port upon which they arrive. Switches in the meshed tree topology share information regarding their directly connected hosts and this information is contained in a virtual SAT or VSAT. Using this information, the paths taken by frames can be optimized because each switch is aware of the switch MT_VID to which an end host is connected. The optimum path can be determined by comparing known MT_VIDs and ports with the VSAT entry. This is possible due to inherent attribute of MT_VIDs. MAC addresses of nodes directly connected to a switch will be learned in much the same way as described in 802.1D; when the hosts generates a frame and it arrives at a non-MTS or *host* port. Ports connecting the switch to a host are the *Host* ports. A port connecting an MTS to another switch participating in the MTP is called an MT port because it is active in the MT topology. Port roles are shown in Fig. 3. Switches populate their tables with addresses from local hosts and map it to their MT_VIDs. They then advertise the Virtual Source Address Table (VSAT) to the neighbors. All switches can exchange VSAT information with their neighbors and add learned information to their own VSAT. This is possible as the MTP does not block ports.

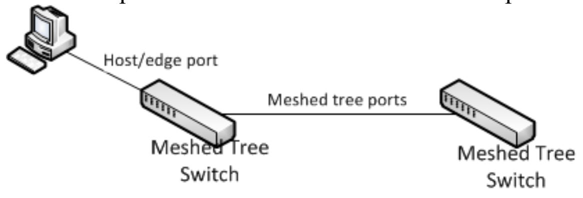


Figure 3. Meshed Tree Switch Port Roles

B. *Messages in MTP*

Switches join a meshed tree topology by either advertising themselves or hearing an advertisement from another MTS. Switches advertise their MT_VIDs using *Hello* messages. While advertising on a particular port they append the port number to their MT_VIDs and offer the MT_VID to a neighbor switch. A switch that accepts an MT_VID from an advertising switch responds with a JOIN message. Switches record the ports on which they hear the join message to retain the child MTS connected on that port. This information is useful in forwarding broadcast frames as described later in this section. The message exchange process is explained with two switches in Fig. 4. Once all switches have at least one MT_VID, the forwarding topology can be viewed as an MT_VID tree. When switches have acquired multiple MT_VIDs, one of these MT_VID trees will be identified as the primary MT_VID (PMT_VID) tree. Unknown MAC addresses, broadcast and multicast traffic will be forwarded via the PMT_VID tree.

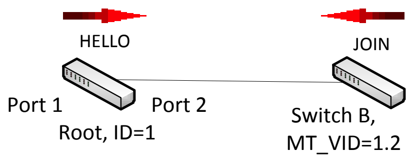


Figure 4. Meshed Tree Hello and Join Process

Once switches have joined the MT topology and understand their parent and child relationships via the MT_VIDs, they exchange information contained in their VSATs via a *VSAT Update* messages (VUM). Upon receipt, the VSAT in the receiving switch is modified in order to provide optimized forwarding to destination host MAC addresses. In more complex topologies, there will be superior pathways between some hosts and these can easily be identified through the MT_VID structure. For example, parent and child switches are direct neighbors and an optimal shortest path will exist unless otherwise defined differently due to path cost.

On discovery of a link failure or other problem, the meshed tree topology responds by deleting MT_VIDs from a switch's MT_VID table and any VSAT entry associated with the lost MT_VID. Because redundant paths are permitted, the topology may have an alternative pathway immediately available. The MT_VID associated with this path may now be elevated to the PMT_VID. Generally speaking, shorter MT_VIDs are preferred as they represent a shorter path, unless the costs of the links define otherwise.

Broadcast Packets: For forwarding broadcast frames or frames to unknown destinations, switches should associate the MT_VIDs to the ports through which they were acquired. Non-root switches forward broadcast frames using the following guidelines; If the broadcast frame is received from the port of PMT_VID, it is sent out on all ports that have an MT_VID derived from the PMT_VID and all host ports. However, if the broadcast frame is received from any other port, it is sent out on ports associated with the PMT_VID and all host ports.

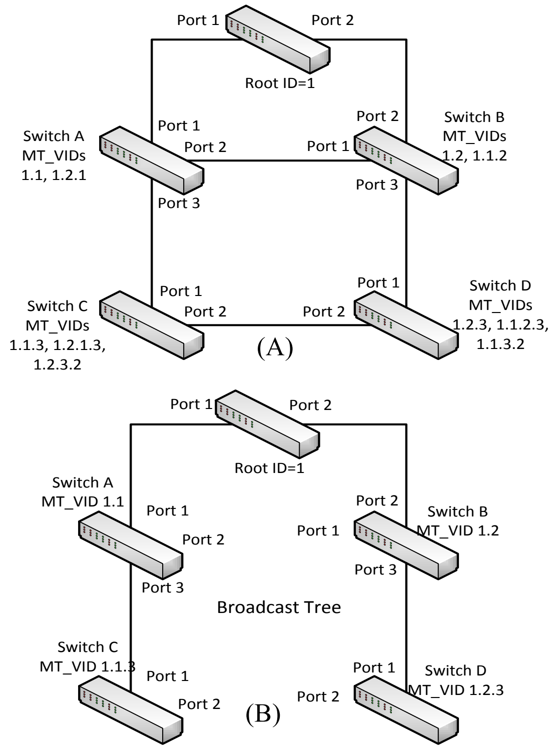


Figure 5. (A) Two Loop Meshed Topology with MT_VIDs. (B) Broadcast Tree for the two-loop Topology

Fig. 5A shows a two-loop topology, which is an extension of Fig. 2 and includes switches C and D. Switches C and D each have acquired three MT_VIDs. Of the multiple MT_VIDs a switch records one MT_VID as the primary MT_VID or PMT_VID. The others are stored in order of preference. The MT_VID tables from all switches are shown in Table II. The shaded MT_VID is the PMT_VID as it has the lowest cost (hops in this case) to the root. In this article all links are assumed to be of equal cost. Based on this information, the PMT_VID tree or broadcast tree is shown in Fig. 5B. In the case of a tie, the MT_VID acquired first would be assumed to be the PMT_VID and this assumption would not impact the operations.

Table II . MT_VID Table at the Switches

Switch	MT_VIDs stored in order of preference
Root	1
A	1.1, 1.2.1
B	1.2, 1.1.2
C	1.1.3, 1.2.1.3, 1.2.3.2
D	1.2.3, 1.1.2.3, 1.1.3.2

IV. OPTIMIZED FORWARDING

All switches that have MT_VIDs populate a VSAT that is indexed by host MAC address. Locally connected hosts are added to the VSAT and in this case the port field is populated with the local switch port. Hosts connected to other switches will be represented in the VSAT with a field listing all of the MT_VIDS of switches that are directly connected to the hosts. This indicates that a VSAT entry for a host may have more than one possible pathway back to the host. For non-local hosts the port field will also contain the egress port for packets destined for that host MAC address. Every time a VSAT entry is changed the forwarding port field is updated to reflect this change. The algorithm used in this case is provided in sub-section B.

If changes were made to the VSAT, the switch creates a new VUM to reflect the changes and multicasts the VUM on all MT ports except the port that received the change. In this way, all of the switches in the topology learn of the VSAT changes.

A. VSAT Update Message

When a host leaves, its VSAT timer expires, or when a new host connects on a port, the switch creates a VSAT Update Message (VUM) and sends the VUM as shown in Fig. 6.

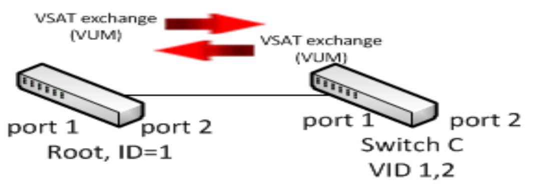


Figure 6. Exchange of Virtual Source Address Tables

A VSAT Update Message (VUM):

- Includes only the changes to the VSAT
- Is sent out on all MT ports using an MT multicast destination address
- Includes host MAC addresses and list of MT_VIDs of the associated switch
- Includes a flag to indicate addition or removal
- Contains a sequence number to avoid duplication of activity and ordering

For each host MAC address in the received VUM, the MTS processes the message as follows:

- If the information is different than an existing VSAT entry; replace if the VUM sequence number if higher
- If not already in the VSAT; add an entry
- If a matching entry exists in the VSAT; do nothing

B. Egress Ports for Frame Delivery

Following cases will be considered to determine the egress port.

Case 1: Destination is this switch, then the egress port is one of the host ports

Case 2: Destination is in this MT branch away from root. Find shortest entry in the forwarding switch's MT_VIDs that is a parent (or grandparent, etc.) to the destination MT_VID. Select the next digit from the MT_VID after the matching pattern; this will be the port to forward the frame.

Case 3: Destination is in this MT branch towards root. Find shortest entry in the forwarding switch's MT_VID for which the destination switch's MT_VID is a parent (or grandparent, etc.). If there is a tie, pick one. Retrieve the port from the VID table; this will be the port to forward the frame.

Case 4: Destination is in a different MT branch off of a switch towards the root. Find an entry in the forwarding switch's VID list that has a common parent (or grandparent, etc.) with the destination switch's MT_VID. This will resolve to the forking switch that leads to the destination. When that switch receives the frame it will use case 3 to direct the frame down the correct branch.

Case 5: Destination is in another MT branch off of the root. This is a special instance of Case 4 where the common parent (or grandparent, etc.) is the root switch. When the root switch gets the frame it will follow case 2 to determine correct branch to send the frame on.

On receiving a VUM, the above process will be executed and the ports associated with the host MAC address can be populated in the VSAT. A typical VSAT entry would be as shown in Fig. 7.

MAC	port	VID
00:01:02:03:04:05	23	1,1 1,2,3

Figure 7. Virtual Source Address Table Entry

V. LINK FAILURE HANDLING BY MTP

Let the link between switches B and D in Fig. 5A fail at time *t*. The time taken by switch D to switch its PVIDs after link failure has been detected is constrained only by the internal hardware / firmware and MTP processing delays. Since MT_VIDs 1123 and 123 were acquired on port 1 of switch D (the port that detected the failure), MT_VID 1132 will take over as the PMT_VID. Switch D then sends a prune message to the switch that has an MT_VID acquired from switch D and derived from the MT_VIDs no longer available.

Switch C continues to use the other paths supported by its other MT_VIDs. In the case of switch B, as it has no MT_VIDs acquired from port 3 (the port that detected link failure) it makes no changes. The broadcast tree after pruning will look as shown in Fig. 8.

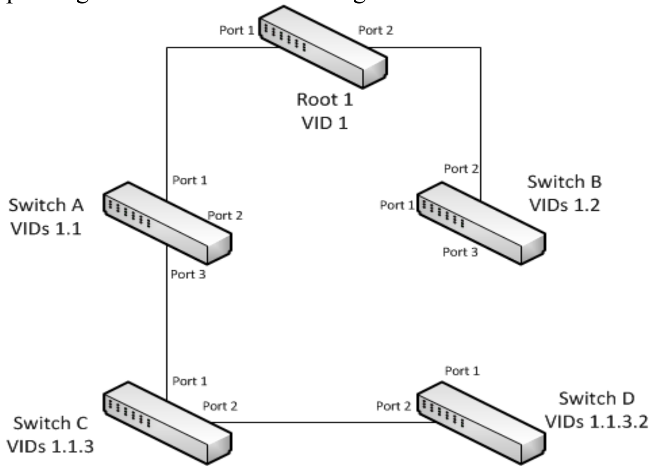


Figure 8. Broadcast Tree After Link Failure

A. Impact on Broadcast Packets:

The switchover of broadcast packets in the midst of PVID tree change will impact the performance. Current schemes do not conisder this. For example: It may happen that some broadcast packet, which should have gone via switch B to switch D may not reach switch D. If the PMT_VID tree at switch C resolves in a timely manner then it may forward the stored or intransit broadcast frames to switch D.

B. Impact on Unicast Packets:

The failover to a backup tree branch after a link failure should occur in near-zero time with MTP. In most cases any performance impact on unicast packets should be negligible. However, there could be cases where frames in transit may experience slight delays. For example a frame from an end node attached to switch C has reached switch D enroute to a node connected to switch B.

Switch D will redirect the frame back on to port 2 towards switch C, which will then forward the frame using another MT_VID. No disruption should occur if the tree pruning and VSAT update occurs before the frame is resent to switch C.

C. Link Failure Process

Two types of messages are used in MTP to detect link failures. The small periodic Hello message containing no information is sent out every 2 seconds to inform the neighbor switch of its continued participation in the MTP. When there are changes in the MT_VID, then *change* Hello messages are sent to inform the neighboring switches of the changes in MT_VID. Timers are used in the switches for the purpose. When the timer for a periodic Hello expires, the switch enters the “Timer Expired” state. The items of relevance are the node’s MT_VIDs and the associated port on the expired link. Following actions are taken.

- VSAT outgoing ports resolved to this port are recalculated and the next best MT_VID of the Destination host MAC address is determined using the algorithm described above.
- Any MT_VID that was received from that switch is flagged as unreachable. VSAT entries for the local hosts are adjusted if required and VSAT updates are sent.
- Since the MT_VID table has now changed, a change_hello packet is sent with the new active MT_VIDs. Downstream nodes will use this information to remove stale MT_VID entries, make corresponding VSAT changes and send VSAT updates.

After the link down event is detected or a timer expires, the first action is the recalculation of best outgoing port. At this point, packets will be forwarded to the correct destination even as the rest of the network heals. While the network converges, some packets may not follow the best possible route, but packet flow will continue. Thus, the convergence time will depend on the failure detection, i.e., the hello timer only. The failover time is almost negligible. When the backup paths can be used without new tree resolution.

VI. SIMULATIONS AND PEFORMANCE

The models for MTP were developed in OPNET. OPNET already had models for RSTP. The performance parameters targeted were the following. Two scenarios were used for the purpose; one with four switches and 1 loop, the other with six switches and 2 loops.

<u>MTP Single Tree Creation (MSTC) Time:</u> this was the time that all switches received at least one MT_VID and can start forwarding frames.

<u>MTP Meshed Tree Creation (MMTC) Time:</u> Each Switch was allowed a maximum of three MT_VIDs. The time taken by all switches to record a maximum of the three different best paths was recorded. In MTP this would be the time when on link failures the backup paths can be used without new tree resolution.

<u>MTP VSAT Update (MVSAT) time:</u> This is the time taken for all switches to record a path to all hosts

subsequent to receiving VUMs. At this time unicast frames can be forwarded without broadcasting.

RSTP initial convergence (IC) time was recorded when the spanning tree was formed. RSTP broadcasts unicast

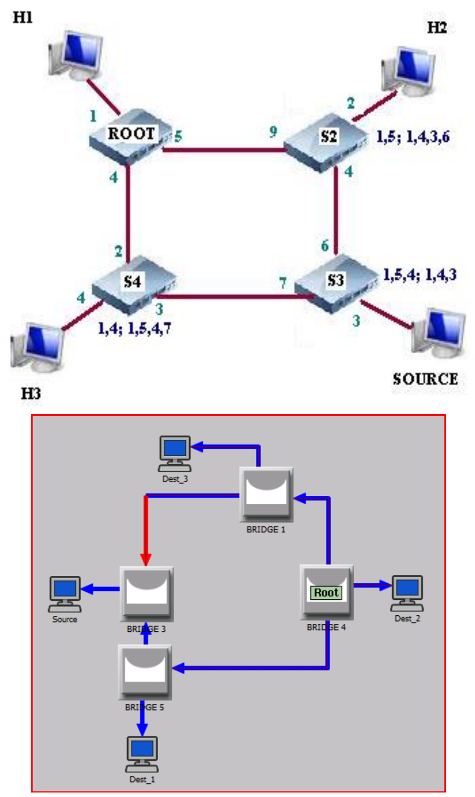


Figure 9. Meshed trees (top), spanning tree (bottom)

The MT_VIDs in Fig. 9 identify the three trees on, which switches S2, S3 and S4 reside. The red line in the picture shows the blocked port in the spanning tree. A host was connected to every switch. One host was identified as the source, which sent packets continuously, while the other hosts sent only for 3 seconds from the start of the simulation. Packet exponential inter-arrival time at the hosts was set to 0.01 sec. At the switches, the control traffic service rate was set to 100,000 packets per sec, while the data traffic service rate was 500,000 packets per sec. Duplex Link speed were maintained at 100 Mbps. Packet sizes were1500 bytes. The duration of simulation was set to 20 secs.

A. 4-Switch Single Loop Scenario

In this scenario, MSTC was recorded as 0.000037 sec, MMTC = 0.000047 sec, while MSAT was 0.0209882 sec. In the case of RSTP, IC was recorded to be 0.55 seconds. In the MTP even if flooding of traffic was avoided during the time that switches learn the host addresses through VUMs, the improvement in convergence is 26 times compared to RSTP. If we allow for frame flooding then the convergence time frames to unknown destinations at this time, as learning time is removed to improve convergence time.

Maximum hops taken by frames.

The resolved topologies for MTP and RSTP in the case of the 4-switch scenario are shown in Fig. 9. improvement is several thousand times. The hops taken by packets in the MTP were recorded to be a maximum of 3 hops. In the case of RSTP the maximum hops would be 4.

Table III. Convergence In MTP – One-Loop Topology

SEED	MSTC	MMTC	MSAT
127	0.000037	0.000047	0.028708
317	0.000037	0.000047	0.007826
509	0.000037	0.000047	0.024935
1009	0.000037	0.000047	0.019308
1721	0.000037	0.000047	0.024164

Note in Table III, for seed 317, the MSAT was as low as 0.007826. The reason for the variance is: when the switch gets the first data packet, it may not have had an MT_VID and hence that packet would have been discarded. The arrival of the second data packet would depend on the seed since the inter-arrival time for data packets is an exponential distribution. So if the second data packet were to trigger VSAT updates from some of the switches, the convergence time would be different for different seeds. Hence, this convergence time depended on the packet inter-arrival at the host. If the inter-arrival were low then the MSAT would be also very low.

B. 6-Switch – Two Loop Scenario

In this scenario, the MSTC, MMTC and MVSAT were recorded to be 0.000047 sec, 0.000070 sec and 0.0225622 seconds as recorded in Table IV. The RSTP IC time was 0.56 seconds. MTP records several thousand times improvement if packets could be forwarded before learning end host addresses and 24 times better after all host addresses were recorded in all switches. The hop counts for packets were recorded to be 6 hops as compared to a maximum of 4 hops with MTP.

The convergence times noted and the hop counts depend on the topology. With more complex and meshed topologies the convergence times and hop counts can vary significantly. For example, in a full meshed topology the maximum hop count for frames in MTP would be 2, whereas for RSTP the frames will have to travel through the root switch. The control message overhead and excess traffic due to frame flooding also would significantly differ.

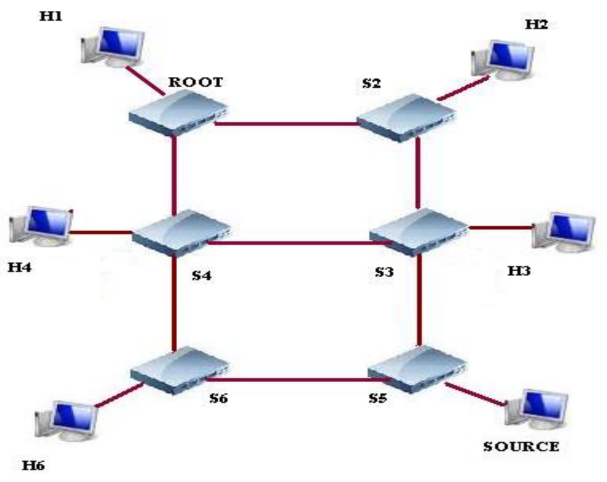


Figure 10. 6-switch, two-loop scenario

Table IV. Convergence In MTP – Two -Loop Topology

SEED	MSTC	MMTC	MSAT
127	0.000047	0.000070	0.033301
317	0.000047	0.000070	0.020941
509	0.000047	0.000070	0.024952
1009	0.000047	0.000070	0.016955
1721	0.000047	0.000070	0.016662

VII. CONCLUSIONS

Loop free forwarding in networks with redundant paths has been previously addressed with the premise that a single logical tree topology originating from a root switch is essential. This resulted in the STA based STP, which had high convergence delays. This was improved by RSTP, which continued to face several disadvantages as stated by the protocol inventors. More complex IS-IS based routing solutions are being adopted at layer 2. This article describes a simple solution that can replace (R)STP at layer 2, without its disadvantages, while at the same time avoid the complexity of using layer 3 routing at layer 2. The specification of the MTP is currently being developed under a new IEEE standard [11].

In this article the MTP performance has been compared with RSTP in terms of convergence times and path hop counts taken by frame traffic. The superior performance achieved with the MTP can be noted from these results. These results can also be used as benchmark when TRILL and SPB are evaluated.

REFERENCES

[1] LAN/MAN Standards committee of the IEEE Computer society, ed. (1998). *ANSI/IEEE Std 802.1D, 1998 Edition, Part 3: Media Access Control (MAC) Bridges*. IEEE Standard.

[2] R. Perlman, D. Eastlake, G. D. Dutt and A. G. Gai, "Rbridges: Base Protocol Specification," RFC 6325, July 2011.

[3] Standard IEEE 802.1Q, Media Access Control Bridges and Virtual Bridged Local Area Networks.

[4] K. Sharma, B. Hartpence, B. Stackpole, D. Johnson and N. Shenoy, "Meshed tree protocol for faster convergence in switched networks," IARIA Sponsored Tenth International Conference on Networking and Services, April 20-24, Chamonix, France, pp 90-95.

[5] N. Shenoy, Y. Pan, D. Narayan, D. Ross and C. Lutzer, "Route robustness of a multi-meshed tree routing scheme for Internet MANETs," Proceeding of IEEE Globecom 2005. 28th Nov to 2nd Dec. 2005 St Louis, pp. 3346-3351.

[6] N. Shenoy and S. Mishra, "Multi-Hop, Multi-Path and Load Balanced Routing in Wireless Mesh Networks," Book Chapter in Encyclopedia on Ad Hoc and Ubiquitous Computing, Published by World Scientific Book Company, 2008.

[7] N. Shenoy, Y. Pan and V.Reddy "Quality of service in Internet MANETs*,"* Proc. of IEEE 16th Intl. Symposium on Personal, Indoor & Mobile Radio Communications (PIMRC), 2005, pp. 1823-1829.

[8] N. Shenoy and Y. Pan. "Multi-meshed tree routing for Internet MANETs," Proc. of 2nd International Symposium on Wireless Communication Systems, pp. 145-149.

[9] S. Pudlewski, N. Shenoy, Y. Al-Mousa, Y. Pan Y and J. Fischer, "A hybrid multi meshed tree routing protocol for wireless ad hoc networks," Second IEEE International Workshop on Enabling Technologies and Standards for Wireless Mesh Networking, September 29, 2008. Atlanta, GA, USA, pp 635-641.

[10] Network Simulation (OPNET Modeler Suite), Riverbed Technologies.

[11] 1910 Working Group for Loop-Free Switching and Routing, Project 1910.1 Standard for Meshed Tree Bridging with Loop Free Forwarding.

[12] W. Wodjek, "Rapid Spanning Tree Protocol: A new solution from old technology,"

http://www.redes.upv.es/ralir/MforS/RSTPtutorial.pdf Reprinted from CompactPCI Systems / March 2003. [Retreived Dec, 2014]

[13] J. Touch and R. Perlman, "Transparent Interconnection of Lots of Links (TRILL): Problem and Applicability Statement," RFC 5556.

[14] R. Perlman, "Rbridges: Transparent Routing," IEEE Proceedings of Infocomm 2004, pp 1211 -1218.

[15] http://mathworld.wolfram.com/DijkstrasAlgorithm.html. [Retreived Dec, 2014]

Appendix A: Computational Complexity of 'Meshed Tree Algorithm'

Assume root is elected, which will not be a consideration if some bridge is already designated to be 'root' or if all bridges would like to set up their own 'meshed tree' as under multi-meshed trees

- We can set a limit on the length of a tree branch and number of 'VIDs' that can be derived from a single bridge without loss of generality. Hence
  a. Let the number of maximum hops in a tree branch from a root node be $\leq B$
  b. Let the number of derived MT_VIDs that each bridge can allocate be $\leq C$
  c. Bridged network size (Number of bridges in the meshed tree) $\leq 1+C^1+C^2+\ldots+C^B$
- The convergence occurs in $N_{iter} = O(1)$ iterations

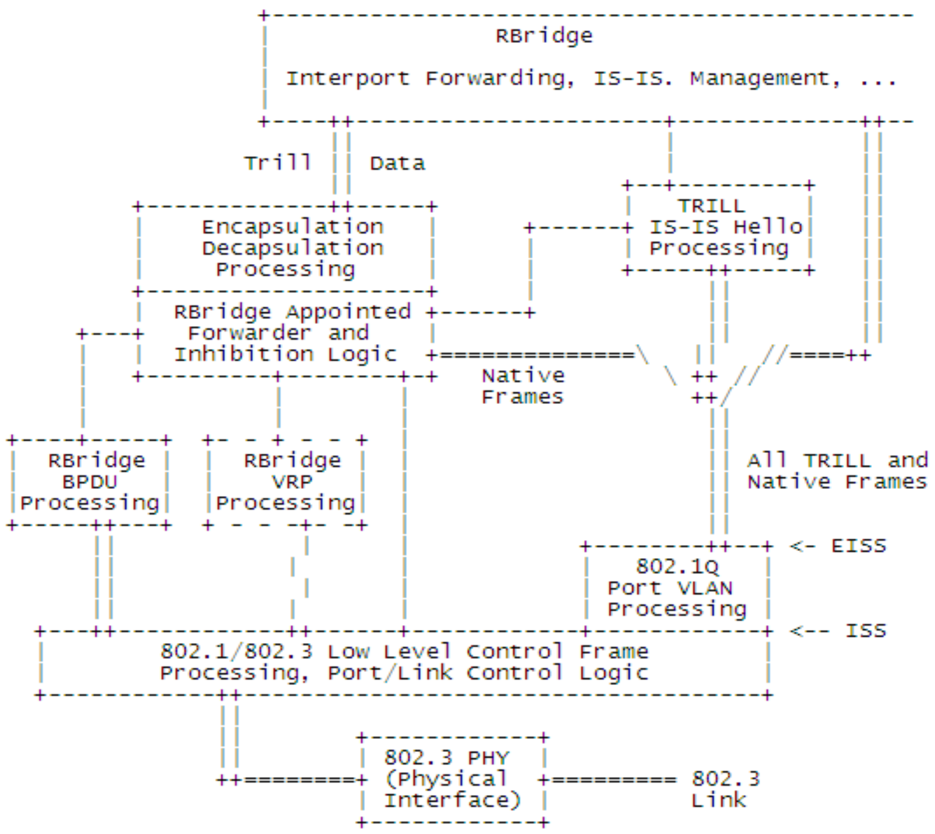
Pseudo Code and Complexity Analysis

This pseudo code is for a bridge attachment to a tree branch in the 'meshed tree' algorithm.

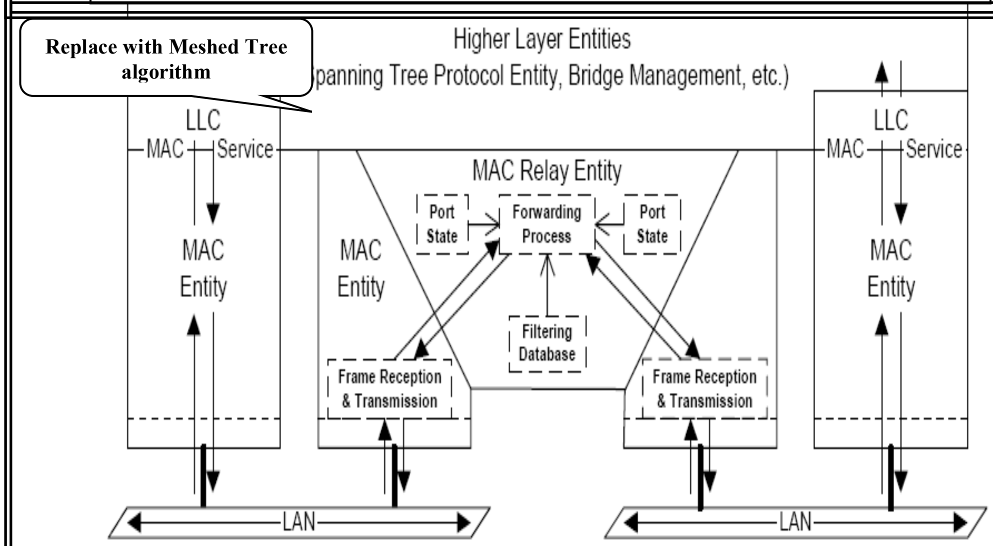
```
Repeat {
If ((hear a 'hello' message)          # a regular 'hello' from my neighbor
                                      # could be a new MT_VID offer
        - Scan the MT_VIDs
        - Compare with my existing MT_VIDs
        - If (new MT_VIDs)
                Repeat for all new MT_VIDs
                { Decision
                Criteria 1: Will the cost be better if I join this MT_VID
                Criteria 2:  Will the hops be within the limit of 'maximum hops'
                Criteria 3:             #any number of other decisions

                Send in a join request for the new MT_VID}
                Else (update the keep_alive timer of my MT_VIDs)
                }
```

As can be seen convergence or decision making iteration is of O(1) on every new MT_VID that is heard.

Appendix B Implementation requirements of 'TRILL on RBridges' and 'Meshed tree on bridges'


RBridge
Interport Forwarding, IS-IS. Management, ...
Trill
Data
Encapsulation Decapsulation Processing
TRILL IS-IS Hello Processing
RBridge Appointed Forwarder and Inhibition Logic
Native Frames
RBridge BPDU Processing
RBridge VRP Processing
All TRILL and Native Frames
802.1Q Port VLAN Processing
<- EISS
<-- ISS
802.1/802.3 Low Level Control Frame Processing, Port/Link Control Logic
802.3 PHY (Physical Interface)
802.3 Link

From http://www.ietf.org/internet-drafts/draft-ietf-trill-rbridge-protocol-11.txt



From IEEE 802.1D "Replaced STA with MTA"

Multicast, TRILL and LISP Extensions for INET

Vladimír Veselý, Marcel Marek, Ondřej Ryšavý and Miroslav Švéda
Department of Information Systems
Faculty of Information Technology, Brno University of Technology (FIT BUT)
Brno, Czech Republic
e-mail: {ivesely, imarek, rysavy, sveda}@fit.vutbr.cz

***Abstract*—Simulation is becoming more important for deploying new technologies or as a proof of concept of new protocols. This paper presents three routing extensions to the INET framework for OMNeT++. The first one is dynamic multicast routing with Protocol Independent Multicast support. The second case is Transparent Interconnection of Lots of Links, which is descendent of data-link layer loop prevention protocols. The third contribution is Locator/ID Split Separation Protocol implementation as currently widely accepted partial solution for Internet scaling crisis.**

Keywords–Multicast; PIM-DM; PIM-SM; TRILL; LISP.

I. INTRODUCTION

The project ANSA (Automated Network Simulation and Analysis) running at the Faculty of Information Technology is dedicated to develop the variety of software tools that can create simulation models based on real networks and subsequently allow for formal analysis and verification of target network configurations. It might be used by public as the routing/switching baseline for further research initiatives using simulator for verification. This paper not only extends our previous work involving multicast routing [1], but also introduces our latest contributions regarding computer networks routing and switching.

Multicast spares network resources, namely bandwidth. Sender and receivers communicate indirectly instead of many separate connections between them. Because of that, multicast traffic is carried across each link only once and the same data is replicated as close to receivers as possible. However, this effectiveness goes concurrently with increased signalization and additional routing information exchange. End-hosts and routers maintain multicast connectivity with the help of following protocols:

- Internet Group Management Protocol (IGMP) [2] / Multicast Listener Discovery (MLD) [3] – End-hosts and first hop multicast-enable routers are using IGMP and MLD protocols for querying, reporting and leaving multicast groups on local LAN segments – they announce their willingness to send or receive multicast data. IPv6 MLD is descendent of IPv4 IGMP, but both protocols are identical in structure and message semantic.
- Distance Vector Multicast Routing Protocol (DVMRP) [4], Multicast Open Shortest-Path First (MOSPF) [5], Protocol Independent Multicast (PIM) – All of them are examples of multicast routing protocols that build multicast topology in router control plane to distribute multicast data among networks. DVMRP and MOSPF are closely tight to the particular unicast routing protocol (RIP, OSPF), whereas variants of Protocol Independent Multicast (PIM) are independent by design and they are using information inside unicast routing table more generally.

The growth of data-centers brings up several problems Data-centers most commonly use Ethernet based networks. Ethernet network provides Layer-2 flat-topology design and with Spanning Tree Protocol (STP) offers seamless plug-and-play approach for connecting new nodes to existing network. The STP guarantees loop-free operation without additional configuration by blocking some ports. The STP was not designed for operation in modern virtualized data-centers and it underutilizes available resources, even though there might be redundant links to the same node or multiple paths to a destination over multiple hops. The STP creates single logical tree to forward unicast and multicast traffic. The RFC 6325 [6] introduced successor called TRILL (Transparent Interconnection of Lots of Links) that treats all these problems. The TRILL accomplishes this by combining functionality of Layer-2 (switching) and Layer-3 (routing). For Layer-3 operation, it takes advantage of slightly modified routing protocol Intermediate System to Intermediate System (IS-IS) [7]. The hardware implementation of TRILL is represented by device called Routing Bridge (RBridge). RBridge's operation is backward compatible with Ethernet Bridging 802.1D and Virtual LAN 802.1Q.

Locator/ID Split Protocol (LISP) development started after IAB Workshop in 2006 as the response dealing with major Internet architecture problems RFC 4984 [8] and follow-up RFC 6227 [9]. IP address functionality is nowadays overloaded; it serves both localization (where) and identification (what) purposes. The main idea behind LISP is to separate those two functions. Then LISP should reduce default-free zone routing table growth, stop prefix deaggregation, allow easier multihoming and mobility without the BGP and split locator and identifier namespaces. LISP supports both IPv4 and IPv6 seamlessly; moreover, it is agnostic to any network protocol. Transition mechanisms are part of the LISP protocol standard, thus it supports communication with legacy non-LISP world.

This paper outlines four simulation modules, which create part of the ANSA project and which extend functionality of the INET framework in OMNeT++.

This paper has the following structure. The next section covers a quick overview of existing OMNeT++ simulation modules relevant to the topic of this paper. Section III describes design of the relevant PIM, TRILL and LISP models. Section IV presents validation scenarios for our implementations. The paper is summarized in Section V together with unveiling our future plans.

II. State of the Art

The current status of support in OMNeT++ 4.5 and INET 2.4 framework is according to our knowledge as follows. We merged functionality of generic IPv4 `Router` and IPv6 `Router6` nodes, so that we created the dual-stack capable router – **ANSARouter**.

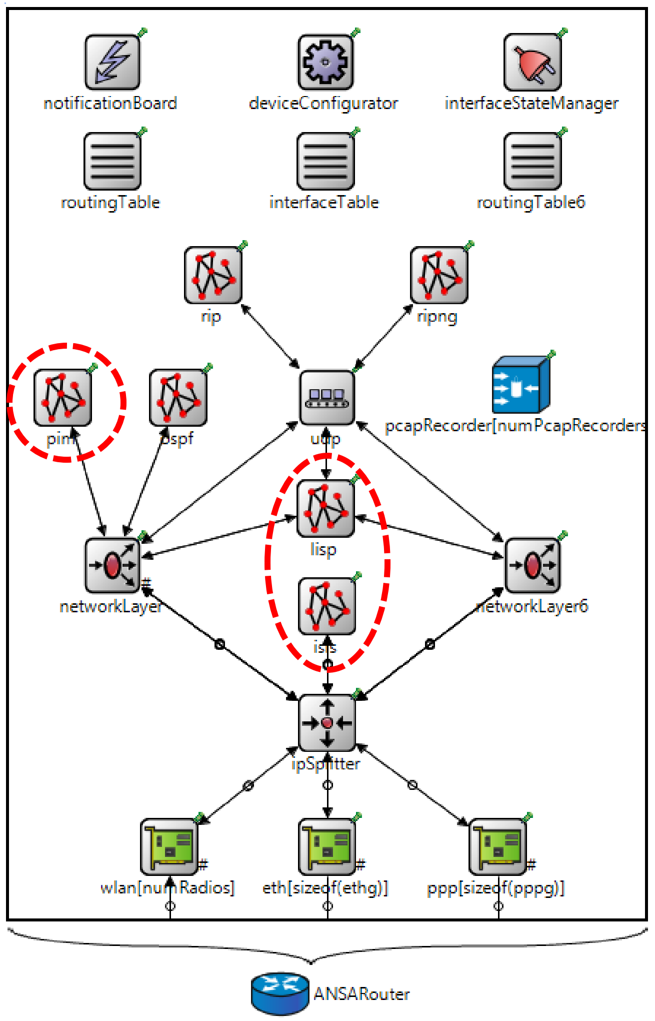


Figure 1. ANSARouter structure with highlighted contribution

We have searched in scientific community around simulation and modeling for other PIM implementations prior to our work. Limited versions (e.g., without *PIM State Refresh* messages) exist for NS-2 [10] or OPNET [11]. However, none of them provides robust implementation (i.e., with finite-state machines implementing whole RFC behavior). Also, existing OMNeT++ multicast attempts proved to be depreciated [12]. We have similarly looked for TRILL, but did not find any project trying to create TRILL simulation implementation. Limited LISP implementation was created [13] to support LISP MobileNode NAT traversal [14]. However, it is intended for the INET-20100323 and OMNeT++ 4.0.

Resulting structure of ANSARouter is in Figure 1 with highlighted simulation modules that are described within this paper.

III. Implementation

A. *PIM – Theory of operation*

All multicast routing protocols provide a function to answer the question, "How to create routing path between sender(s) and receivers?" Baselines for this functionality are distribution trees of the following two types:

Source trees – The separate shortest path tree is built for each source of multicast data. A sender is the root and receivers are the leaves. However, memory and computation overhead causes this type is not scalable in the case of a network with many sources of multicast. In these situations usually the Shared tree is used.

Shared trees – A router called **Rendezvous Point (RP)** exists in a topology that serves as a meeting point for the traffic from multiple sources to reach destinations. The shared tree interconnects RP with all related receivers.

There are four PIM operational modes: PIM Dense Mode (PIM-DM), PIM Sparse Mode (PIM-SM), Bidirectional PIM (BiDir-PIM) and PIM Source-Specific Multicast (PIM-SSM). All of them differ in signalization, employed distribution trees and suitable applications.

Multicast routing support is performed by one dedicated router on each LAN segment elected based on *PIM Hello* messages. This router is called **designated router (DR)** and it is the one with the highest priority or highest IP address.

PIM-DM is recommended for topologies with only one multicast source and lots of receivers. PIM-DM can be easily deployed without burdening configuration on active devices. However, PIM-DM does not scale well when number of sources increases. For this situation or for topologies with sparsely connected receivers, PIM-SM is suggested to be employed. Sparse mode scales much better in large topologies comparing to Dense mode, but configuration and administration is more complicated. PIM-SSM suits for multicast groups containing multiple sources providing the same content where client using IGMPv3 or MLDv2 may specify from which particular source it wants to receive data. BiDir-PIM is intended for topologies where many-to-many communication occurs. Currently, PIM-DM and PIM-SM are widely deployed PIM variants. Hence, we decided to implement them as the first.

PIM-DM idea consists of initial data delivery to all multicast-enable destinations (to flood multicast traffic everywhere), where routers prune themselves explicitly from the distribution tree if they are not a part of the multicast group. PIM-DM is not taking advantage of RP; thus, it is using source trees only.

PIM-DM routers exchange following messages during operation:

- *PIM Hello* – Used for neighbor detection and forming adjacencies. It contains all settings of shared parameters used for DR election;
- *PIM Prune/Join* – Sent towards upstream router by downstream device to either explicitly prune a source tree, or to announce willingness to receive multicast data by another downstream device in case of previously solicited *PIM Prune*;
- *PIM Graft* – Sent from a downstream to an upstream router to join previously pruned distribution tree;
- *PIM Graft-Ack* – Sent from an upstream to a downstream router to acknowledge *PIM Graft*;
- *PIM State Refresh* – Pruned router refreshes prune state upon receiving this message;
- *PIM Assert* – In case of multi-access segment with multiple multicast-enabled routers one of them must be elected as an authoritative spokesman. Mutual exchange of *PIM Asserts* accomplishes this election.

On the contrary to PIM-DM, **PIM-SM** works with different principle where initially no device wants to receive multicast. Thus, all receivers must explicitly ask for multicast delivery and then routers forward multicast data towards end-hosts. PIM-SM employs both types of multicast distribution trees. Sources of multicast are connected with RP by source trees – source of multicast is the root of a source tree. RP is connected with multicast receivers by shared trees – RP is the root of shared tree. Multicast data is traversing from sources down by source tree to RP and further down by shared tree to receivers. PIM-SM cannot work properly until all PIM routers in a network do not know exactly which router is RP for a given multicast group.

PIM-SM exchanges subsequent message types:

- *PIM Hello* – same as PIM-DM.
- *PIM Register* – Sent by source's DR towards RP whenever new source of multicast is detected.
- *PIM Register-Stop* – Solicited confirmation of *PIM Register*. It is sent by RP in reverse direction that source's DR can stop registering process of a new source. RP is aware of multicast data and may send them to receivers via shared tree.
- *PIM Prune/Join* – This message forms the shape of source and shared distribution trees. Multiple sources could provide data to the same multicast group – each one of them sends data via own source tree towards RP, from here data is reflected to receivers via shared tree.
- *PIM Assert* – same as PIM-DM.

The thorough survey on PIM-DM and PIM-SM message exchange scenarios are out of scope of this paper. More can be found in RFC 3973 [15] and RFC 4601 [16]; let us state that our implementations (i.e., finite-state machines, message structure, etc.) fully comply with IETF's standards.

B. PIM – Design

We have synthetized multiple finite-state machines that describe behavior of PIM-DM and PIM-SM with reference to used timers and exchanged PIM messages [17]. Figure 2 shows implemented architecture of the `pim` module.

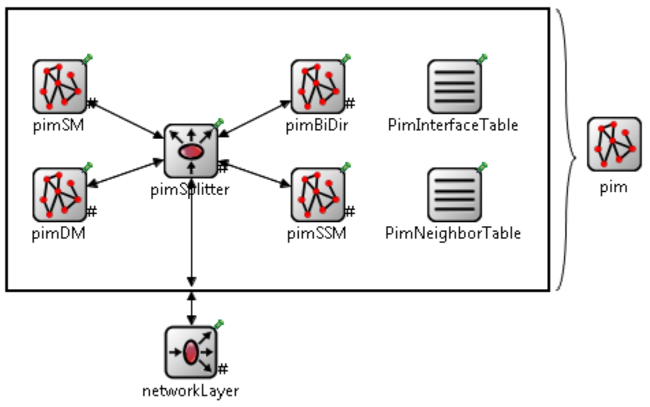


Figure 2. Proposed PIM module design

Besides previous modules, there were also some minor alternations to IPv4 `networkLayer` as well as to IPv4 `routingTable` module.

Implementation is done in NED (model design) and C++ (model behavior) languages. Brief description of implemented components is summarized in Table I.

TABLE I. DESCRIPTION OF PIM SUBMODULES

Name	Description
`pimSplitter`	This submodule is connected with INET `networkLayer`. It inspects all PIM messages and passes them to appropriate PIM submodules.
`pimDM`	The main implementation and logic of PIM-DM protocol is over here.
`pimSM`	The main implementation and logic of PIM-SM protocol is over here.
`pim InterfaceTable`	Stores all PIM relevant information for each router's interface.
`pim NeighborTable`	Keeps state of formed PIM adjacencies and information about neighbors (PIM version they are using, priorities, neighbors IPs).
`pimSSM, pimBiDir`	Prepared as a placeholder for upcoming implementations of BiDir-PIM and PIM-SSM variants.

C. TRILL – Theory of operation

TRILL provides loop-less topology for Layer-2. It replaces obsolescent STP protocol. Devices that actually run TRILL are called **RBridges**. The work of RBridge can be divided into two separate components – routing and switching.

The first component is based on link state approach and employs so called **IS-IS Layer-2** implementation. All RBridges run instance of this altered IS-IS to exchange link-state statuses (LSPs) for the whole topology. The IS-IS Layer-2 instance uses single IS-IS Level-1 area with zero-length Area-ID, which contains all RBridges as if they are in one large flat Layer-2 network. **Designated RBridge** (**DRB**) is elected from the set of RBridges on shared link and it chooses Appointed Forwarder (AF) for this link. DRB informs others about chosen AF via *TRILL Hello* messages. **Appointed Forwarder** acts as ingress and egress gate to the **campus** (area covered by single TRILL instance). **Designated VLAN** carries all TRILL-encapsulated traffic between RBridges.

DRB is in charge of appointing Designated VLAN. Encapsulated frame format is depicted in Figure 3.

The second component is TRILL itself. The TRILL distinguishes five classes of traffic:

- *TRILL L2 Control* – Frames of low level Layer-2 protocols like STP (e.g., BPDUs). TRILL control frames are processed locally.
- *Native* – Non-TRILL traffic from/to hosts. Only AF sends and receives native traffic on shared segment.
- *TRILL Data* – TRILL encapsulated frames with Ethernet's header field *Ethertype* set to 0x22F3.
- *TRILL Control* – Frames that belongs to Layer-2 IS-IS protocol. They have *Ethertype* set to 0x22F4 value.
- *TRILL other* – Other frames, which do not match any of the previous types, are dropped without acknowledgment.

RBridge distinguishes between three port types:

- *Access Port* – handles native non-TRILL traffic from hosts and delimits campus edges.
- *Trunk Port* – handles TRILL Data frames. These ports are located inside campus.
- *Hybrid Port* – handles both previous traffic types. This port interconnects partitioned campus across non-TRILL area.

New outer Ethernet header	**TRILL header**	**Original unchanged Ethernet header**
Ethernet Payload		**FCS**

Figure 3. TRILL frame encapsulation

Native frame is equipped with TRILL header (see Figure 4) as soon as it passes first RBridge. Additionally, outer Ethernet header is also prepended. Subsequently, either whole encapsulated frame is sent towards egress RBridge that has destination host connected, or native frame is forwarded on local port. **Multi-destination frame** is used when destination is unknown. RBridge sends this kind of frame: a) in native form on all links where this RBridge acts as an AF; b) as TRILL encapsulated to its neighbors according to given distribution tree. RBridge learns the source MAC address each time it receives frame in native form on the port for which this RBridge is AF.

Version (2b)	**Reserved** (2b)	**M** (1b)
Op-Length (5b)	**Hop Count** (6b)	
Egress RBridge Nickname (16b)		
Ingress RBridge Nickname (16b)		
Options…		

Figure 4. TRILL header format

When RBridge receives TRILL encapsulated frame, it either sends it toward egress RBridge according to `RBMACTable` (unicast case), or to all connected branches of a given distribution tree (multi-destination case). If the receiving RBridge is also the egress RBridge then the frame is decapsulated and sent to the local port.

Distribution Trees are used when sending multi-destination frames. RBridge with the highest priority in campus decides about the number of distribution trees and their roots.

Complete description of TRILL protocol is out of scope of this document.

D. *TRILL – Design*

We created a new RBridge simulation model. This model comprises existing IS-IS module that has been extended by IS-IS Layer-2 design and plug-and-play configuration-less functionality. The complete structure is shown in Figure 5.

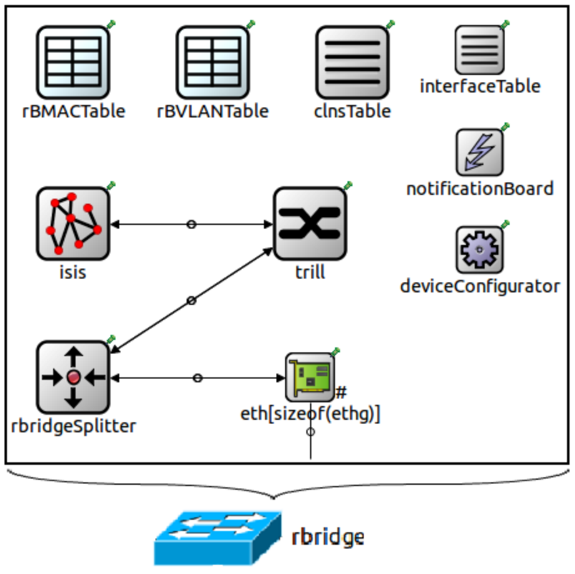


Figure 5. Proposed RBridge module design

The overview of each submodule is given in Table II.

TABLE II. DESCRIPTION OF RBRIDGE SUBMODULES

Name	Description
`RBEthInterface`	This module handles MAC part of the INET `EthernetInterface` module without de/encapsulation.
`RBridgeSplitter`	This submodule acts as a placeholder for future integration with other modules.
`ISIS`	Submodule contains L2/L3 version of IS-IS routing protocol. An appropriate version is chosen based on device type.
`TRILL`	The main implementation and logic.
`RBMACTable`	It plays similar role as does routing table for Layer-3 protocols. It resolves destination to a set of output ports.
`RBVLANTable`	It stores information about active VLANs (i.e., name, VLAN ID and associated ports).
`clnsTable`	Submodule stores next-hop addresses to all accessible destinations via routes with the best metric. It supports load balancing for routes with equal metric.

As a first step, we needed to change behavior of `RBEthInterface` module to use only the MAC part, but leave the de/encapsulation to TRILL. After the incoming frame passes through `RBEthInterface`, it is delivered to

`TRILL` module. Then the frame is classified and processed based on previously mentioned traffic class. For unicast frames, the sender is learned and put into `RBMACTable`.

Every RBridge generates Distribution trees independently based on its link-state database content.

E. LISP – Theory of operation

LISP accomplishes loc/id separation by splitting the IP address into two namespaces:

- **Routing Locator (RLOC)** namespace with addresses fulfilling their localization purposes by telling where device is connected in the network.
- **Endpoint Identifier (EID)** namespace where each device has unique name that distinct it from each other.

There is (and probably always will be) a non-LISP namespace where direct LISP communication is (even intentionally) not supported. Apart from namespaces exist also: a) specialized routers performing map-and-encap that interconnects different namespaces; b) dedicated devices maintaining mapping system; c) proxy routers allowing communication between LISP and non-LISP world.

LISP mapping system performs lookups where a set of RLOCs is retrieved for a given EID. Following map-and-encap principle, original (inner) header is encapsulated by a new (outer) header, which is appended when crossing borders from EID to RLOC namespace. Whenever packet is crossing back from RLOC to EID namespace, packet is decapsulated by stripping off outer header. LISP places additional UDP header succeeded by LISP header between inner and outer header. LISP uses reserved port numbers – 4341 for data and 4342 for signalization traffic. Currently any combination of IPv4/v6 headers is supported.

Basic components are **Ingress Tunnel Router (ITR)** and **Egress Tunnel Router (ETR)**. Both are border devices between EID and RLOC space, the only difference is in which direction they are operated. The single device could be either ITR only, or ETR only, or ITR and ETR at the same time. Usually, the functionality is dual and we denote this kind of device with abbreviation **xTR**.

ITR is the exit point from EID space (a.k.a. LISP **site**) to RLOC space, which encapsulates original packet. This process may consist of querying mapping system followed by updating local **map cache** where EID-to-RLOC mapping pairs are stored for limited time to reduce signalization overhead.

ETR is the exit from RLOC space to EID space, which decapsulates original header. This means that outer header plus auxiliary UDP and LISP headers are stripped off. ETR is also announcing all LISP sites (their EID addresses) and by which RLOCs they are accessible.

LISP mapping system is primary employing two components – **Map Resolver (MR)** and **Map Server (MS)**. Looking for RLOC to EID is analogous process as DNS name resolution. In case of DNS, host asks its DNS resolver (configured within OS) which IP address belongs to a given fully qualified domain name. DNS server responds with cached answer or delegates the question recursively or iteratively to another DNS server according to the name hierarchy. In case of LISP, the querier is ITR that needs to find out, which RLOCs could be used to reach a given EID. ITR has preconfigured MR, which is bothered each time mapping is needed. Mapping queries are data-driven. This means that data transfer between LISP sites initiates mapping process and data itself is postponed until mapping is discovered. Map cache on each ITR holds only those records that are actively needed by ongoing traffic.

Following list contains all LISP mapping signalization messages with their brief description. LISP control traffic are LISP packets without inner header – just outer header + UDP header with source and destination ports set on 4342 + appropriate LISP message header. Structural details of each message can be found in RFC 6830 [18].

- *LISP Map-Register* – Each ETR announces as authority one or more LISP sites to the MS employing this message. Each registration contains a list of RLOCs to a given EID with properties.
- *LISP Map-Request* – ITR generates this request whenever it needs to discover current EID-to-RLOC mapping and sends it into mapping system.
- *LISP Map-Reply* – This is solicited response from the mapping system to a previous request and contains all RLOCs to a certain EID together with their attributes. Each ITR has its own map cache where information from replies are stored for a limited time and used locally to reduce signalization overhead of mapping system.
- *LISP Negative Map-Reply* – Mapping system generates this message as a response whenever given identifier is not the EID and thus proxy routing for non-native LISP communication must occur.

MR accepts LISP Map-Requests sent by ITR. Message is either delegated further into mapping system (namely to appropriate MS), or MR responds with LISP Negative Map-Reply if questioned EID is address from non-LISP world.

Every MS maintains **mapping database** of LISP sites that are advertised by LISP Map-Register messages. If MS receives LISP Map-Request then: a) either MS responds directly to querying ITR – it is allowed to do that because MS has all the necessary information in its mapping database; b) or MS forwards request towards designated ETR that is successfully registered to MS for target EID.

Each RLOC record to a given EID has two attributes – priority and weight. **Priority** (one byte long value in range from 0 to 255) expresses each RLOC preference. The locator with the lowest priority is used by ITR when creating outer header. Communication may be load-balance based on **weight** (in range from 0 to 100) between multiple RLOCs sharing the same priority. Priority value 255 means that locator must not be used for traffic forwarding. Zero weight means that RLOC may be used for load-balancing according to ITR wishes.

F. LISP – Design

LISP xTR, MR and MS functionality is currently implemented within `LISPRouting` compound module that is interconnected with both (IPv4) `networkLayer` and (IPv6) `networkLayer6`. It consists of three submodules that are depicted in the Figure 6 and described in Table below the figure.

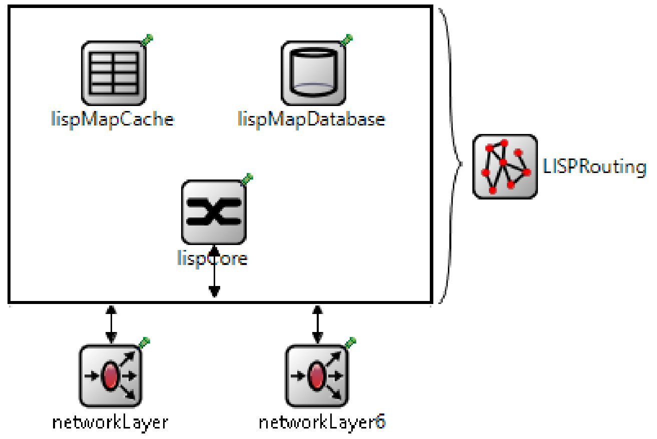


Figure 6. Proposed LISPRouting module design

TABLE III. DESCRIPTION OF LISPROUTING SUBMODULES

Name	Description
`lispCore`	The heart that is responsible for handling LISP control and data traffic. It independently combines functionality of ITR, ETR, MR and MS. In case of ITR, this involves encapsulation and active maintenance of map cache. In case of ETR, it is responsible for decapsulation process and site registration. In case of MR, it simply delegates map queries. In case of MS, it maintains mapping database.
`lispMapCache`	Local LISP map cache that is populated on demand by routing data traffic between LISP sites. Each record (EID-to-RLOC mapping) has its own separate handling (i.e., expiration, refreshment, availability of RLOCs).
`lispMapDatabase`	MS's mapping database that maintains LISP site registration by ETRs. It contains site specific information (e.g., shared key, statistics of registrars and times of registration). Each site also contains known EID-to-RLOC mappings.

Minor changes were done also to both `networkLayer/6` submodules in order to divert LISP data traffic intended for encapsulation/decapsulation towards `LISPRouting` module (UDP port 4341). `LISPRouting` is also registered with `UDPSocket` on local port 4342 to handle LISP control messages coming from UDP submodule.

IV. TESTING

In this section, we provide information on testing and validation of our implementations using several test scenarios. We have built exactly the same topologies for both simulation and real network and observed (using transparent switchport analyzers and packet sniffers) relevant messages exchange between devices.

For multicast operation, we compared the results with the behavior of referential implementation running at Cisco routers (Cisco 2811 routers with IOS operating system version c2800nm-advipservicesk9-mz.124-25f) and host stations (with FreeBSD 8.2 OS).

For TRILL, we did a comparison only with specifications.

As for LISP, we conducted tests and compared them to a referential behavior of Cisco routers (C7200 routers with IOS c7200-adventerprisek9-mz.152-4.M2) and host stations (with Windows 7 OS).

A. PIM-DM

We had considered multiple different topologies and decided for one which is just enough large to test every multicast aspect and still with scenario easy to follow. In this testing network (topology is shown in Figure 7), we have three routers (`R1`, `R2` and `R3`), two sources of multicast (`Source1` and `Source2`) and three receivers (`Host1`, `Host2` and `Host3`).

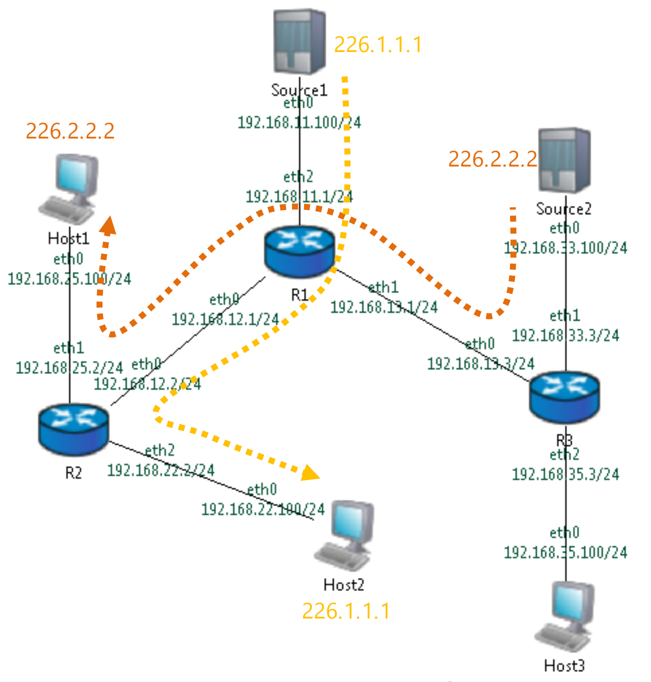


Figure 7. PIM-DM testing topology

We scheduled actions covering all phases of multicast communication (i.e., sources start and stop sending and hosts start and stop receiving of multicast data). Scheduled scenario is summarized in Table IV.

TABLE IV. PIM-DM EVENTS SCENARIO

Phase	Time [s]	Device	Multicast action	Group
#1	0	`Host1`	Starts receiving	226.2.2.2
#2	87	`Source1`	Starts sending	226.1.1.1
#3	144	`Host2`	Starts receiving	226.1.1.1
#4	215	`Source2`	Starts sending	226.2.2.2
#5	364	`Host2`	Stops receiving	226.1.1.1
#6	399	`Source2`	Stops sending	226.2.2.2

Hosts sign themselves to receive data from particular multicast group via *IGMP Membership Report* message during phases #1 and #3. Similarly, the host uses *IGMP Leave*

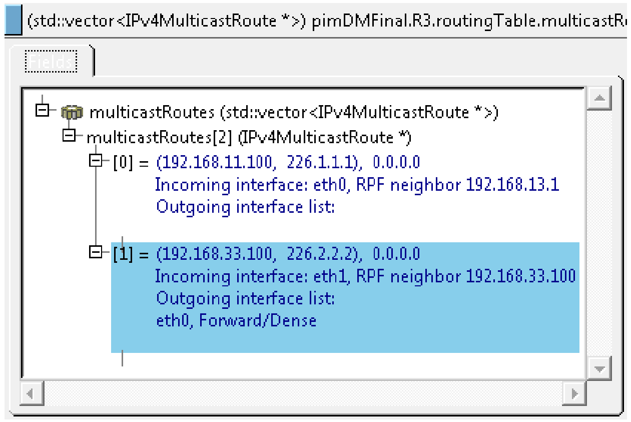
Group message to stop receiving data during phases #5 and #6.

#1) There are no multicast data transferred. Only *PIM Hellos* are sent between neighbors.

#2) First multicast data appears but, because of no receivers, routers prune themselves from source distribution tree after initial flooding.

#3) `Host2` starts to receive data from group 226.1.1.1 at the beginning of #3. This means that `R2` reconnects to source tree with help of *PIM Graft*, which is subsequently acknowledged by *PIM Graft-Ack*.

#4) The new source starts to send multicast data. All routers are part of the source distribution tree with `R3` as the root. `R3` acts as RP that is illustrated in Figure 8.



```
(std::vector<IPv4MulticastRoute *>) pimDMFinal.R3.routingTable.multicastR
multicastRoutes (std::vector<IPv4MulticastRoute *>)
  multicastRoutes[2] (IPv4MulticastRoute *)
    [0] = (192.168.11.100, 226.1.1.1), 0.0.0.0
        Incoming interface: eth0, RPF neighbor 192.168.13.1
        Outgoing interface list:

    [1] = (192.168.33.100, 226.2.2.2), 0.0.0.0
        Incoming interface: eth1, RPF neighbor 192.168.33.100
        Outgoing interface list:
        eth0, Forward/Dense
```

Figure 8. R3 multicast routing table after phase #4

#5) `Host2` is no longer willing to receive multicast from 226.1.1.1 and, because `Host2` is also the only listener to this group, `R2` disconnects itself from distribution tree with *PIM Prune/Join*.

#6) Finally `Source2` stops sending data to the group 226.2.2.2 at the beginning of #6. Subsequent to this, no PIM message is generated. Routers just wait for 180 seconds and then wipe out an affected source tree from the multicast routing table.

The message confluence proved correctness of our PIM-DM implementation by simulation as well as by real network monitoring, which can be observed in Table V.

TABLE V. TIMESTAMP COMPARISON OF PIM-DM MESSAGES

Phase	Message	Sender	Simul. [s]	Real [s]
#1	*PIM Hello*	R1	30.435	25.461
#2	*PIM Prune/Join*	R3	87.000	87.664
#3	*PIM Graft*	R2	144.000	144.406
	PIM Graft-Ack	R1	144.000	144.440
#5	*PIM Prune/Join*	R2	366.000	364.496

B. *PIM-SM*

For testing purposes of PIM-SM, topology is more complex. We have two designated routers (`DR_R1`, `DR_R2`) for receivers (`Receiver1`, `Receiver2`), two DRs (`DR_S1`, `DR_S2`) for sources (`Source1`, `Source2`) and one rendezvous point (`RP`). The scenario is depicted in Figure 9.

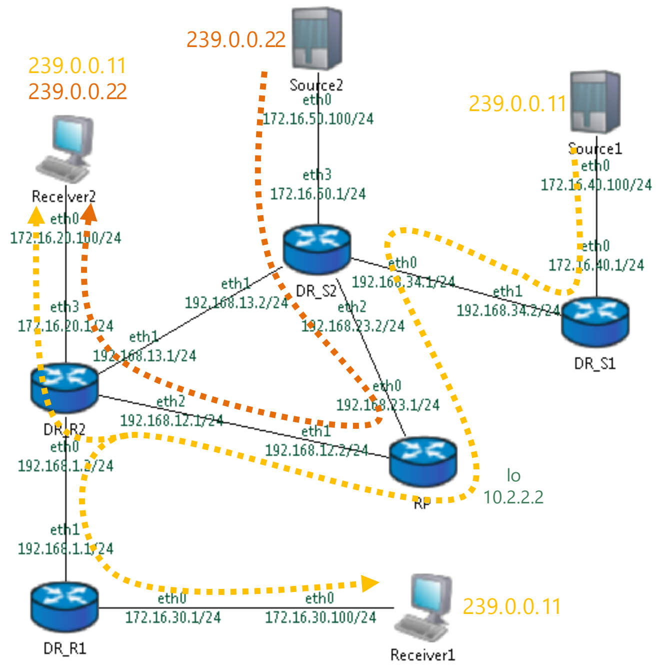


Figure 9. PIM-SM testing topology

A scenario for PIM-SM is summarized in Table VI and additional description of actions follows bellow.

TABLE VI. PIM-SM EVENTS SCENARIO

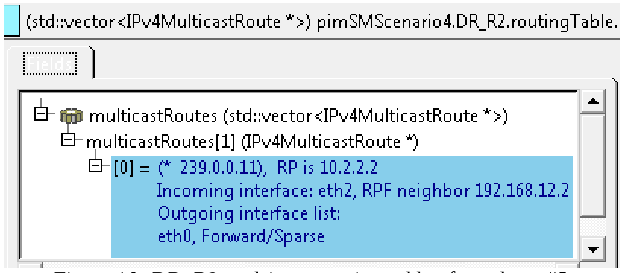
Phase	Time [s]	Device	Multicast action	Group
#1	10	`Source1`	Starts sending	239.0.0.11
#2	20	`Receiver1`	Starts receiving	239.0.0.11
#3	25	`Receiver2`	Starts receiving	239.0.0.11
#4	40	`Receiver2`	Starts receiving	239.0.0.22
#5	60	`Source2`	Starts sending	239.0.0.22
#6	90	`Receiver1`	Stops receiving	239.0.0.11
#7	120	`Receiver2`	Stops receiving	239.0.0.11
#8	220	`Receiver2`	Stops receiving	239.0.0.22
#9	310	`Source1`	Stops sending	239.0.0.11
#10	360	`Source2`	Stops sending	239.0.0.22

Just as in PIM-DM scenario, receivers send *IGMP Membership Report* and *IGMP Leave Group* messages to sign on and off the multicast groups during phases #2, #3 and #6-#8.

#1) `Source1` starts to send multicast data. Those data is encapsulated into *PIM Register* message sent by `DR_S1` via `DR_S2` towards `RP`. Following next `RP` responds with *PIM Register-Stop* back to `DR_S1`, thus registration of new source is finished.

#2) *IGMP Membership Report* for multicast group 239.0.0.11 by `Receiver1` turns on joining process of `DR_R1` and `DR_R2` to shared tree and joining of `RP` and `DR_S2` to source tree by sending *PIM Join/Prune*.

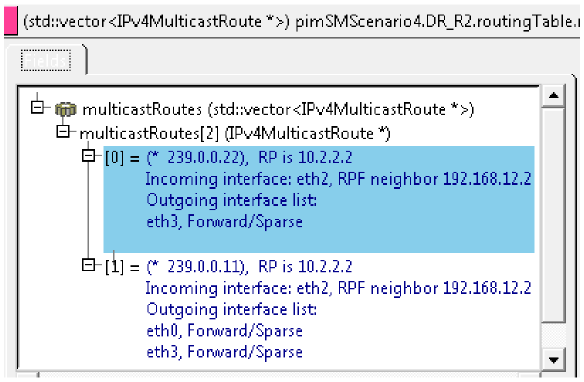
#3) `DR_R2` is already connected to a shared tree, thus *IGMP Membership Report* only adds another outgoing interface to shared tree as could be seen in Figure 10.



```
(std::vector<IPv4MulticastRoute *>) pimSMScenario4.DR_R2.routingTable.
Fields
multicastRoutes (std::vector<IPv4MulticastRoute *>)
  multicastRoutes[1] (IPv4MulticastRoute *)
    [0] = (* 239.0.0.11), RP is 10.2.2.2
        Incoming interface: eth2, RPF neighbor 192.168.12.2
        Outgoing interface list:
        eth0, Forward/Sparse
```

Figure 10. DR_R2 multicast routing table after phase #3

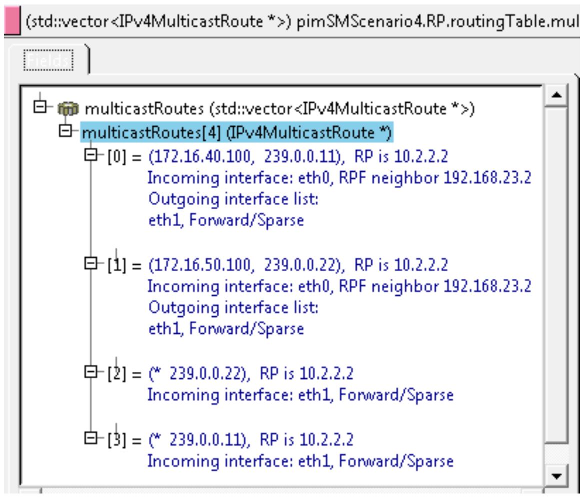
#4) Whenever `Receiver2` starts receiving multicast group 239.0.0.22, new multicast route is added on `DR_R2` (see Figure 11). Subsequently `DR_S2` joins to shared tree via *PIM Join/Prune* sent towards `RP`.



```
(std::vector<IPv4MulticastRoute *>) pimSMScenario4.DR_R2.routingTable.
Fields
multicastRoutes (std::vector<IPv4MulticastRoute *>)
  multicastRoutes[2] (IPv4MulticastRoute *)
    [0] = (* 239.0.0.22), RP is 10.2.2.2
        Incoming interface: eth2, RPF neighbor 192.168.12.2
        Outgoing interface list:
        eth3, Forward/Sparse

    [1] = (* 239.0.0.11), RP is 10.2.2.2
        Incoming interface: eth2, RPF neighbor 192.168.12.2
        Outgoing interface list:
        eth0, Forward/Sparse
        eth3, Forward/Sparse
```

Figure 11. DR_R2 multicast routing table after phase #4

#5) `Source2` starts sending multicast data to 239.0.0.22 after `Receiver1` already joined the shared tree. `DR_S2` registers source with *PIM Register* that contains also multicast data. These data is decapsulated and sent down via shared tree to receivers. As a next step, `RP` joins the source tree via *PIM Prune/Join* message and a moment later it confirms registration via *PIM Register-Stop* sent towards `DR_S2`. Multicast routes on `RP` converged and they could be observed in Figure 12.



```
(std::vector<IPv4MulticastRoute *>) pimSMScenario4.RP.routingTable.mul
multicastRoutes (std::vector<IPv4MulticastRoute *>)
  multicastRoutes[4] (IPv4MulticastRoute *)
    [0] = (172.16.40.100, 239.0.0.11), RP is 10.2.2.2
        Incoming interface: eth0, RPF neighbor 192.168.23.2
        Outgoing interface list:
        eth1, Forward/Sparse

    [1] = (172.16.50.100, 239.0.0.22), RP is 10.2.2.2
        Incoming interface: eth0, RPF neighbor 192.168.23.2
        Outgoing interface list:
        eth1, Forward/Sparse

    [2] = (* 239.0.0.22), RP is 10.2.2.2
        Incoming interface: eth1, Forward/Sparse

    [3] = (* 239.0.0.11), RP is 10.2.2.2
        Incoming interface: eth1, Forward/Sparse
```

Figure 12. RP multicast routing table after phase #5

#X) Every 60 second after successful source registration, the given DR and RP exchange empty *PIM Register* and *PIM Register-Stop* messages to confirm presence of multicast source. Also every 60 seconds after last receiver joined multicast group, PIM router refreshes upstream connectivity to any tree via *PIM Prune/Join* message. This phase cannot be planned or scheduled; it is default behavior of PIM-SM protocol finite-state machine. It is illustrated only once for `Source1` distribution trees but the same message exchange happens also for `Source2`.

#6) Upon receiving *IGMP Leave Group*, `DR_R1` prunes itself from shared tree via *PIM Prune/Join* message sent upstream to `DR_R1`. `DR_R1` then removes interface `eth0` as outgoing interface for multicast group 239.0.0.11.

#7) `Receiver2` decides not to receive multicast from `Source1`. Its *IGMP Leave Group* starts pruning process that goes from `DR_R2` up to `DR_S1`. On each interim PIM router, multicast route for 239.0.0.11 is removed via *PIM Prune/Join* message.

#8) Later `Receiver2` signs off from receiving 239.0.0.22, which causes similar exchange of *PIM Prune/Join* as in case of #7.

#9) Whenever `Source1` stops sending multicast, elimination process starts for a given multicast route. As time goes by, ExpireTimer times out on every PIM router and multicast distribution tree for 239.0.0.11 is wiped out from routing table. The same approach applies for #10.

Validation testing against the real-life topology shows just reasonable time variations (around ±3 seconds). This variation observable on real Cisco devices is caused by two factors: a) control-plane processing delay; b) stochastic message jitter to avoid potential race conditions in similar processes. Table VII outlines results.

TABLE VII. TIMESTAMP COMPARISON OF PIM-SM MESSAGES

Phase	Message	Sender	Simul. [s]	Real [s]
#1	*PIM Register*	`DR_R1`	10.005	10.127
	PIM Register-Stop	`RP`	10.006	10.380
#2	*PIM Prune/Join*	`DR_R1`	20.001	20.422
	PIM Prune/Join	`DR_R2`	20.002	20.813
	PIM Prune/Join	`RP`	20.003	21.117
	PIM Prune/Join	`DR_S2`	20.005	21.320
#4	*PIM Prune/Join*	`DR_R2`	40.001	43.524
#5	*PIM Register*	`DR_S2`	60.000	61.459
	PIM Prune/Join	`RP`	60.003	61.970
	PIM Register-Stop	`RP`	60.004	62.758
#X	*PIM Register*	`DR_S1`	70.008	74.304
	PIM Register-Stop	`RP`	70.009	75.671
	PIM Prune/Join	`DR_R1`	80.000	83.041
	PIM Prune/Join	`DR_R2`	80.001	83.647
	PIM Prune/Join	`RP`	80.003	83.950
	PIM Prune/Join	`DR_S2`	80.003	84.004
#6	*PIM Prune/Join*	`DR_R1`	90.000	92.909
#7	*PIM Prune/Join*	`DR_R2`	120.001	122.311
	PIM Prune/Join	`RP`	120.002	122.704
	PIM Prune/Join	`DR_S2`	120.003	123.296

C. TRILL

TRILL testing topology consists of six RBridges and two stations (`Host1` with IP 172.16.30.100 and `Host2` with 172.16.3.0.101) as depicted in Figure 14. Both stations belong to VLAN 1. CLNS address plan is in Table IX.

TRILL scenario includes network convergence to stable state and sending ICMP Echo Request/Reply messages (ping) between two hosts. Each RBridge gradually builds up its routing table (`clnsTable`) via IS-IS process and generates distribution trees for each RBridge in topology.

TABLE VIII. TRILL EVENTS SCENARIO

Phase	Time [s]	Device	Action
#1	0	RB*	Start sending *TRILL Hello*
#2	5	RB*	Start generating and sending LSPs
#3	10	Host1	Sends *ARP Request*
#4	10	Host2	Sends *ARP Reply*
#5	10	Host1	Sends *ICMP Echo Request*
#6	10	Host2	Sends *ICMP Echo Reply*

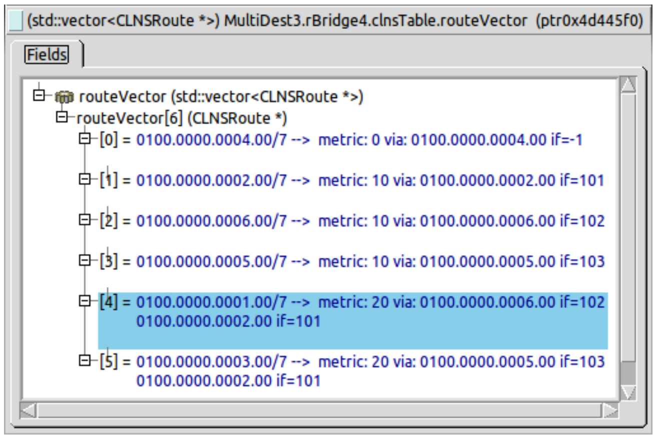
TABLE IX. DEVICE CONFIGURATION

Device	Address
RB1	0100.0000.0001
RB2	0100.0000.0002
RB3	0100.0000.0003
RB4	0100.0000.0004
RB5	0100.0000.0005
RB6	0100.0000.0006

The list of important phases (summarized in Table VIII) for TRILL verification scenario follows down below:

#1) All RBridges start sending TRILL hello messages in simulation time `t=0s` to discover their neighbors. All neighborships converge to Report state after 2.8 seconds.

#2) At time `t=5s` RBridges generate and send LSPs to neighbors. Topology is completely converged at a time `t=5.9s` and each RBridge initiate shortest-path first algorithm to fill up `clnsTable`. The content of this table for `RB4` is depicted in Figure 13. Highlighted line shows two equal cost paths to `RB1` with metric 20.



```
(std::vector<CLNSRoute *>) MultiDest3.rBridge4.clnsTable.routeVector  (ptr0x4d445f0)
Fields
routeVector (std::vector<CLNSRoute *>)
  routeVector[6] (CLNSRoute *)
    [0] = 0100.0000.0004.00/7 -->  metric: 0 via: 0100.0000.0004.00 if=-1
    [1] = 0100.0000.0002.00/7 -->  metric: 10 via: 0100.0000.0002.00 if=101
    [2] = 0100.0000.0006.00/7 -->  metric: 10 via: 0100.0000.0006.00 if=102
    [3] = 0100.0000.0005.00/7 -->  metric: 10 via: 0100.0000.0005.00 if=103
    [4] = 0100.0000.0001.00/7 -->  metric: 20 via: 0100.0000.0006.00 if=102
          0100.0000.0002.00 if=101
    [5] = 0100.0000.0003.00/7 -->  metric: 20 via: 0100.0000.0005.00 if=103
          0100.0000.0002.00 if=101
```

Figure 13. RB4's clnsTable

#3) In simulation time `t=10s`, `Host1` sends *ARP Request* with broadcast MAC address. `RB1` learns `Host1`'s MAC address from received frame and store it in `RBMACTable`. `RB1` encapsulates *ARP Request* frame with TRILL header and sends on all interfaces in its distribution tree. `RB1`'s distribution tree includes interfaces to `RB2` and `RB6`. Every RBridge, which received this frame, learns source MAC address of the inner frame. The frame similarly propagates through the rest of the network until it is decapsulated and sent to `Host2`, because `RB5` is AF on that link.

#4) `Host2` replies with *ARP Reply* to `Host1`'s MAC address. `RB5` looks up this MAC address in its `RBMACTable` for egress RBridge address. Then the `RB5` queries his `clnsTable` to get next-hop RBridge address and output interface. Encapsulated *ARP Reply* frame is now handled as a unicast frame throughout the network. This response travels through `RB5`, `RB4`, and `RB6` to `RB1`, where it is decapsulated and send to `Host1`. This process is illustrated in Figure 14.

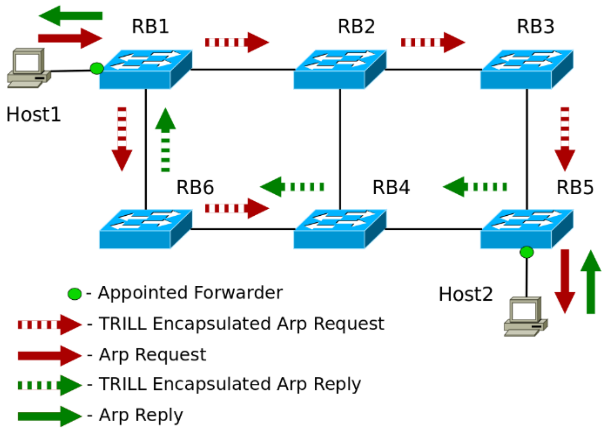


Figure 14. ARP messages propagation

#5) `Host1` finally sends *ICMP Echo Request* with resolved `Host2` MAC address. `RB1` already knows egress RBridge for `Host2`'s MAC address from received *ARP Reply*. The *ICMP Echo Request* is prepended with TRILL header across campus. `RB5` acts as an egress RBridge and therefore decapsulates received frame and sends it to `Host2`.

#6) `Host2` generates *ICMP Echo Reply*. It is again encapsulated. However, it travels through different path on the way back to `Host1` as shown in Figure 15.

TRILL testing topology was verified against proposed behavior of RFC 6325 specifications. Its conformation with other existing implementations (for instance on Cisco or HP routers) is subject of further verification process.

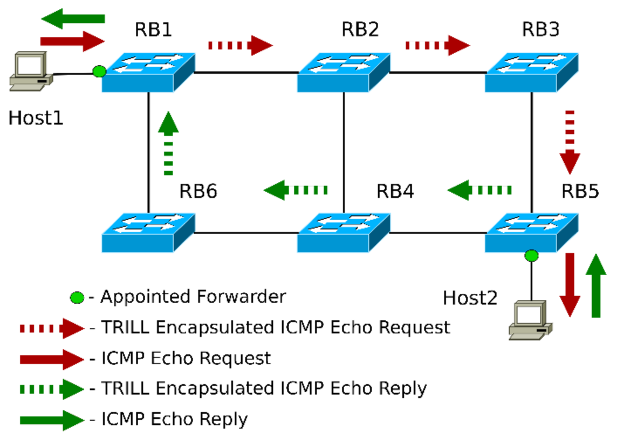


Figure 15. ICMP messages propagation

D. *LISP*

We have verified LISP implementation on the topology depicted in Figure 16. It contains two sites (bordered by XTRs `xTR_A` and `xTR_B`). The topology contains router `MRMS`, which acts as MR and MS for both sites. IPv4 only capable core is simulated by a single `Core` router. Static routing is employed to achieve mutual connectivity across core.

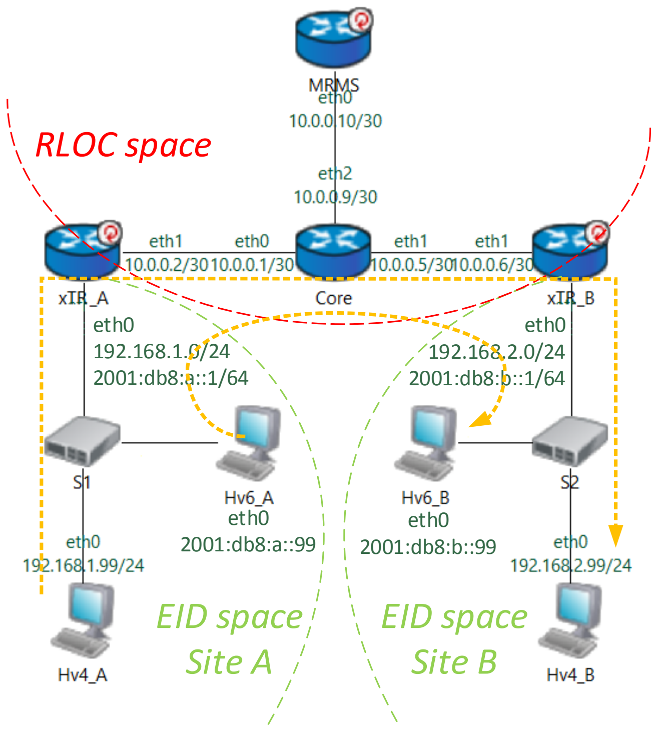
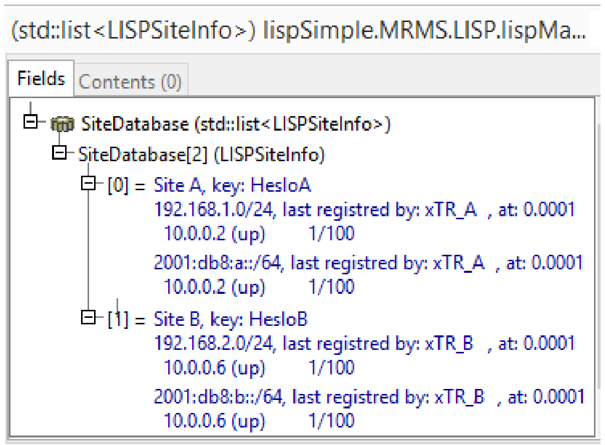


Figure 16. LISP testing topology

For this test, we configured `xTR_*` to register EID-to-RLOC mappings by its MS (which is `MRMS`). We scheduled *ICMP Request/Reply* between IPv4 only hosts `Hv4_A` and `Hv4_B` and same for IPv6 only `Hv6_A` and `Hv6_B`. Scenario beginning (phase #1 at `t=0s`) is aligned at the time of the first *ICMP Echo Request* from `Hv4_A`. Start of Phase #1 is aligned with successful site registration in phase #0.

Test goes through following phases:

- #0) First `xTR_A` and `xTR_B` must register their EID-to-RLOC mappings to its MS (`MRMS`). This means that for each mapping is generated *LISP Map-Register* message (with destination 10.0.0.10) and it is periodically resent every 60 seconds in order to keep mapping liveliness. Correctly populated `MRMS`'s mapping database is depicted in Figure 17.



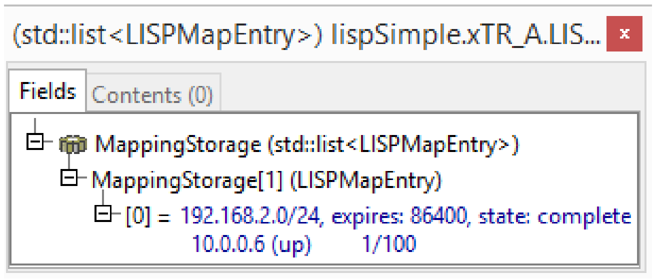
(std::list<LISPSiteInfo>) lispSimple.MRMS.LISP.lispMa...

Fields | Contents (0)

```
⊟ SiteDatabase (std::list<LISPSiteInfo>)
  ⊟ SiteDatabase[2] (LISPSiteInfo)
    ⊟ [0] = Site A, key: HesloA
        192.168.1.0/24, last registred by: xTR_A , at: 0.0001
         10.0.0.2 (up)      1/100
        2001:db8:a::/64, last registred by: xTR_A , at: 0.0001
         10.0.0.2 (up)      1/100
    ⊟ [1] = Site B, key: HesloB
        192.168.2.0/24, last registred by: xTR_B , at: 0.0001
         10.0.0.6 (up)      1/100
        2001:db8:b::/64, last registred by: xTR_B , at: 0.0001
         10.0.0.6 (up)      1/100
```

Figure 17. MRMS's mapping database

- #1) `Hv4_A` initiates *ICMP Echo Request* with `Hv4_B`'s IPv4 address as destination. Packet is received by `xTR_A` and the destination is treated as EID from other LISP site. Hence, `xTR_A` generates *LISP Map-Request* for 192.168.2.1/32 that is sent to `MRMS`.
- #2) `MRMS` receives `xTR_A`'s mapping query from. Subsequently, `MRMS` checks its mapping database in order to find proper ETR for requested address. EID is part of registered "Site B". The query is forwarded to `xTR_B` because of that.
- #3) `xTR_B` receives *LISP Map-Request* and responds with *LISP Map-Reply* that tells the `xTR_A` that available RLOC is 10.0.0.6.
- #4) `xTR_A` receives reply and cache the answer into the local map cache (see Figure 18). This mapping has default expiration time of 1440 minutes.



(std::list<LISPMapEntry>) lispSimple.xTR_A.LIS...

Fields | Contents (0)

```
⊟ MappingStorage (std::list<LISPMapEntry>)
  ⊟ MappingStorage[1] (LISPMapEntry)
    ⊟ [0] = 192.168.2.0/24, expires: 86400, state: complete
        10.0.0.6 (up)      1/100
```

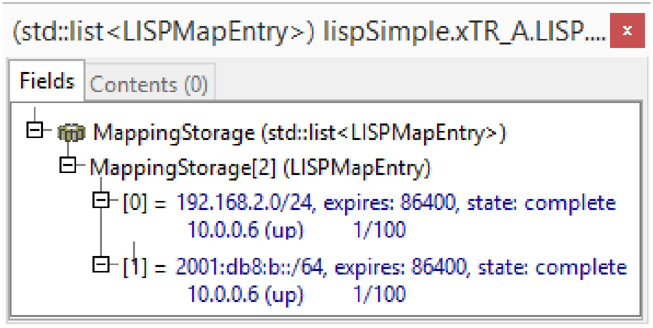
Figure 18. xTR_A's map cache after #4

- #5) Due to the fact that mapping is known, `xTR_A` wraps a new outer IPv4 header (10.0.0.2 as the source and 10.0.0.6 as the destination RLOCs) around any *ICMP Echo Request* and forwards it out through `eth1`. Core routes the packet to `xTR_B` where it is decapsulated and forwarded to its destination (Hv4_B). Hv4_B responds with *ICMP Echo Reply*.

#6) Whenever xTR_B receives xTR_A's query, xTR_B starts its own reverse-mapping process to determine EID-to-RLOC mapping for requesting EID 192.168.1.1/32. xTR_B generates *LISP Map-Request* that is sent to MRMS and from here passed to xTR_A, which answers with *LISP Map-Reply*. The result is inserted as a new record into xTR_B's map cache.

#7) Later Hv6_A initiates *ICMPv6 Echo Request* towards Hv6_B. xTR_A receives packet, which starts *LISP Map-Request* for EID 2001:db8:b::9/128.

From this point, behavior is same as in phases #2-6 with slight difference that now IPv6 traffic (ICMP replaced by ICMPv6) is carried across (IPv4 only) core. The final content of xTR_A map cache is shown in Figure 19.


(std::list<LISPMapEntry>) lispSimple.xTR_A.LISP....

Fields | Contents (0)

MappingStorage (std::list<LISPMapEntry>)
MappingStorage[2] (LISPMapEntry)
[0] = 192.168.2.0/24, expires: 86400, state: complete
10.0.0.6 (up) 1/100
[1] = 2001:db8:b::/64, expires: 86400, state: complete
10.0.0.6 (up) 1/100

Figure 19. xTR_A's map cache after #13

We have compared behavior of simulated and real network just as in the case of PIM testing process. The results for phases from #1 to #6 are summarized in Table X.

TABLE X. TIMESTAMP COMPARISON OF LISP MESSAGES

Phase	Message	Sender	Simul. [s]	Real [s]
#1	*ICMP Echo Request*	Hv4_A	0.000 drop	0.000 drop
	LISP Map-Request	xTR_A	0.000	0.249
#2	*LISP Map-Request*	xTR_A	0.002	0.278
#3	*LISP Map-Request*	MRMS	0.004	0.318
#4	*LISP Map-Reply*	xTR_B	0.005	0.459
#5	*ICMP Echo Request*	Hv4_A	1.000 2.000	1.113 drop 2.478
	ICMP Echo Reply	Hv4_B	1.003 2.003	2.527
#6	*LISP Map-Request*	xTR_B	0.005	1.301
	LISP Map-Request	MRMS	0.007	1.528
	LISP Map-Reply	xTR_A	0.009	1.542

ICMP Echo Requests are dropped due to the missing EID-to-RLOC mapping that becomes available after *LISP Map-Reply* in phase #6. Therefore, there is one (resp. two) ICMP packet drop(s) in case of simulated (resp. real) network.

There are slight variations due to the same reasons as in case of PIM validation. Messages in the simulator are emitted based on atomic event scheduler. However, events in real router are dispatched according to CPU interrupts and availability of hardware resources, which may create additional delays comparing to processing in simulator. However, overall routing outcome (i.e., LISP sites connectivity, local map cache content) is the same comparing simulation and real hardware when applying timescale perspective with second's precision.

V. CONCLUSION AND FUTURE WORK

In this paper, we discussed options for dynamic multicast routing, loc/id split and data-link loop-prevention. We presented an overview of currently existing modules relevant to above topics in OMNeT++. The main contributions are simulation models for PIM-DM, PIM-SM, TRILL and LISP that extend functionality of our ANSARouter and overall INET framework. Also, we introduce simulation scenarios and their results, which show that our implementations comply with relevant RFCs.

We plan to wrap up our native multicast implementation by adding IPv6 support. For TRILL, we intend to include dynamic nickname negotiation together with MTU discovery and VLAN mapping detection. Furthermore, we plan to simulate proposed LISP improvement, which should synchronize map caches. This may lead to shortening of lookup times and better performance in high availability scenarios.

More information about ANSA project is available on webpage [19]. Source codes of simulation modules could be downloaded via GitHub repository [20].

ACKNOWLEDGMENT

This work was supported by the Brno University of Technology organization and by the research grants:

- FIT-S-14-2299 supported by Brno University of Technology;
- VG20102015022 supported by Ministry of the Interior of the Czech Republic;
- IT4Innovation ED1.1.00/02.0070 supported by Czech Ministry of Education Youth and Sports.

REFERENCES

[1] V. Veselý, O. Ryšavý, and M. Švéda, "Protocol Independent Multicast in OMNeT++," The Tenth International Conference on Networking and Services, pp. 132-137 Chamonix, France, 2014.

[2] B. Cain, S. Deering, I. Kouvelas, B. Fenner, and A. Thyagarajan, "RFC 3376: Internet Group Management Protocol, Version 3," October 2002. [Online]. Available: https://tools.ietf.org/html/rfc3376. [Accessed: February 2014].

[3] R. Vida and K. Costa, "RFC 3810: Multicast Listener Discovery Version 2 (MLDv2) for IPv6," June 2004. [Online]. Available: http://tools.ietf.org/html/rfc3810. [Accessed: February 2014].

[4] D. Waitzman, C. Partridge, and S. Deering, "RFC 1075: Distance Vector Multicast Routing Protocol," November 1988. [Online]. Available: http://tools.ietf.org/html/rfc1075 [Accessed: February 2014].

[5] J. Moy, "RFC 1584: Multicast Extensions to OSPF," March 1994. [Online]. Available: http://tools.ietf.org/html/rfc1584. [Accessed: February 2014].

[6] R. Perlman, "RFC 6325: Routing Bridges (RBridges): Base Protocol Specification," July 2011. [Online]. Available: http://tools.ietf.org/html/rfc6325.

[7] ISO/IEC 10589:2002, "Information technology - Telecommunications and information exchange between systems - Intermediate System to Intermediate System intra-domain routeing information exchange protocol for use in conjunction with the protocol for providing the CLNS," October 2002. [Online]. Available: http://standards.iso.org/ittf/PubliclyAvailableStandards/c030932_ISO_IEC_10589_2002(E).zip.

[8] D. Meyer, L. Zhang, and K. Fall, "RFC 4984: Report from the IAB Workshop on Routing and Addressing," September 2007. [Online]. Available: http://tools.ietf.org/html/rfc4984.

[9] T. Li, "RFC 6227: Design Goals for Scalable Internet Routing," May 2011. [Online]. Available: http://tools.ietf.org/html/rfc6227.

[10] T. Henderson, "31. Multicast Routing," November 2011. [Online]. Available: http://www.isi.edu/nsnam/ns/doc/node338.html. [Accessed: November 2014].

[11] C. Adam and Y. Chien, "Implementing the Protocol Independent Multicast (Sparse Mode) on the Opnet Simulation Platform," May 1998. [Online]. Available: http://www.cs.columbia.edu/~hgs/teaching/ais/1998/projects/pim/report.html. [Accessed: November 2014].

[12] R. Leal, J. Cacinero, and E. Martin, "New Approach to Inter-domain Multicast Protocols," ETRI Journal, vol. 33, no. 3, pp. 355-365, June 2011.

[13] D. Klein, M. Hoefling, M. Hartmann, and M. Menth, "Integration of LISP and LISP-MN into INET," in Proceedings of the IEEE 5th International ICST Conference on Simulation Tools and Techniques, Desenzano del Garda, Italia, 2012.

[14] D. Klein, M. Hartmann, and M. Menth, "NAT Traversal for LISP Mobile Node," July 2010. [Online]. Available: http://tools.ietf.org/html/draft-klein-lisp-mn-nat-traversal.

[15] A. Adams, J. Nichols, and W. Siadak, "RFC 3973: Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised)," January 2005. [Online]. Available: http://tools.ietf.org/html/rfc3973.

[16] B. Fenner, M. Handley, H. Holbrook, and I. Kouvelas, "RFC 4601: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)," August 2006. [Online]. Available: http://tools.ietf.org/html/rfc4601.

[17] V. Rybová and T. Procházka, "Protocol Independent Multicast: Finite-state machines for Dense and Sparse Mode," Brno University of Technology, October 2013. [Online]. Available: https://nes.fit.vutbr.cz/ansa/uploads/Main/pim-fsm.pdf. [Accessed: January 2014].

[18] D. Farinacci, V. Fuller, D. Meyer, and D. Lewis, "RFC 6830: The Locator/ID Separation Protocol (LISP)," January 2013. [Online]. Available: http://tools.ietf.org/html/rfc6830.

[19] Brno University of Technology, ANSAWiki | Main / HomePage, January 2014. [Online]. Available: http://nes.fit.vutbr.cz/ansa/pmwiki.php. [Accessed: November 2014].

[20] GitHub, December 2013. [Online]. Available: https://github.com/kvetak/ANSA. [Accessed: November 2014].

Design of a Novel Network Architecture for Distributed Event-Based Systems Using Directional Random Walks in an Ubiquitous Sensing Scenario

Cristina Muñoz and Pierre Leone
Department of Computer Science
University of Geneva
Carouge, Switzerland
Email: {Cristina.Munoz, Pierre.Leone}@unige.ch

Abstract—**Ubiquitous sensing devices frequently disseminate data among them. The use of a distributed event-based system that decouples publishers from subscribers arises as an ideal candidate to implement the dissemination process. In this paper, we present a network architecture that merges the network and overlay layers of typical structured event-based systems. Directional random walks are used for the construction of this merged layer. Our strategy avoids using a specific network protocol that provides point-to-point communication. This implies that the topology of the network is not maintained, so that nodes not involved in the system are able to save energy and computing resources. We evaluate the performance of the overlay layer using directional random walks and pure random walks for its construction. Our results show that directional random walks are more efficient because: (1) they use less nodes of the network for the establishment of the active path of the overlay layer and (2) they have a more reliable performance. Furthermore, as the number of nodes in the network increases, so do the number of nodes in the active path of the overlay layer for the same number of publishers and subscribers. Finally, we discard any correlation between the number of nodes that form the overlay layer and the maximum Euclidean distance traversed by the walkers.**

***Keywords*-Distributed Event-Based Systems; Overlay Layer; Directional Random Walks; Pure Random Walks; Wireless Sensor Networks.**

I. Introduction

Ubiquitous or pervasive computing [1][2] uses many sources and destinations to gather and process data related to physical processes with the aim of making possible human-computer interaction. In the process of dissemination, some devices generate the data, while others are waiting for the sensing data. In this context, the use of a distributed event-based system [3] arises as an ideal candidate to implement the model of communication on the reception or transmission of events.

The main characteristic of an event-based system is that publishers and subscribers are decoupled. This means that they do not have any information about each other. The element in charge of matching notifications with subscriptions is called the event notification service. In distributed networks, the event notification service is implemented using a network of brokers nodes (see Figure 1). It is considered that a broker is any node in the network that has information about any single or set of subscriptions. The complexity of designing this type of systems usually lies on the way to elect the nodes that act as brokers because of the decentralized nature of a distributed network.

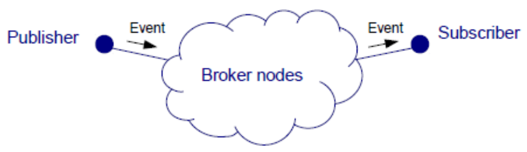


Figure 1: Distributed notification service using a network of brokers.

In our research [1], we assume that a node can be a publisher, a subscriber, a broker or a combination of these three possibilities. We also assume that all the nodes in the network are able to participate in it without the requirement to adopt the specific role of publisher or subscriber. Nodes that are actively participating in the network but do not take any specific role will be considered as part of the overlay layer. Those nodes of the overlay layer that are able to redirect messages will be considered as brokers.

Event-based systems are classified as topic-based or content-based [3]. Topic-based systems take into account the subject of messages in order to match publications with subscriptions. Content-based systems use filters to specify the value of subscriptions attributes to redirect notifications. A filter is a boolean function that depends on the set of subscriptions. In our proposal, we plan to deal with a content-based system that uses Bloom filters at broker nodes in order to save memory resources and speed up routing decisions.

Sensor networks frequently use tiny devices with limited battery capabilities that make unsuitable the use of a Global Positioning System (GPS) to disseminate information according to the coordinates of nodes. In addition to this, the use of virtual coordinates to substitute real coordinates requires the use of sinks or landmarks to structure the network. For these reasons, the use of coordinates in an unstructured sensing scenario is not recommended. We assume that we work in an unstructured scenario in which no routing protocol provides communication between the nodes of the network.

The constraints of the network infrastructure lead us to the design of a network architecture for distributed event-based systems that must use as less resources as possible (i.e., battery, memory, etc.). In this paper, we present a solution that avoids implying all the nodes of the network in the dissemination process by using a distributed notification service defined by Directional Random Walks (DRWs).

The rest of this paper is organized as follows: Section II analyzes the state of the art. Section III points out the approach to solve the problem specified in this section. Section IV presents the research efforts already done for the approach specified in Section III. Section V details the process of construction of the proposed architecture. Section VI evaluates the performance of our solution using DRWs, comparing it with the use of Pure Random Walks (PRWs). Finally, Section VII summarizes our proposal.

II. State of the Art

A. *Distributed and Structured Event-based Systems*

Distributed and structured event-based systems use three layers on the top of a bottom layer (see Figure 2), which provides data link functionalities, to facilitate topology control:

1) The network layer is in charge of providing data forwarding between the different nodes involved in the network. A network protocol, such as the Multicast Ad-hoc On-demand Distance Vector (MAODV) [4] is needed to provide point-to-point communication.
2) The medium layer is called the overlay layer. It is a virtual layer that builds the event notification service by providing a network of brokers that redirect notifications to the corresponding subscribers.
3) Finally, on the top layer the event-based protocol is implemented.

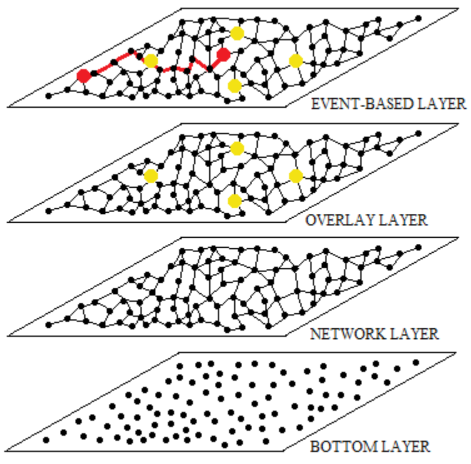


Figure 2: Decomposition in layers of the typical design of a distributed and structured event-based system.

One strategy to construct the overlay layer is to use a tree. In TinyMQ [5], which is designed specifically for wireless sensor networks, a multi-tree overlay layer is maintained.

Another strategy is to clusterize the network and use cluster heads to manage messages as in Mires [6], which is a middleware for sensor networks. The Gradient Landmark-Based Distributed Routing (GLIDER) [7] organizes the network using some defined landmarks to compute the Delaunay graph for network partition. Then, the Landmark-Based Information Brokerage scheme (LBIB) [8] uses an overlay layer based in GLIDER to match publishers with subscribers.

A typical solution is to build the overlay layer using Distributed Hash Tables (DHTs). In these systems, a key is mapped to a particular node with storage location properties. In some DHT architectures, rendez-vous nodes depend on the node ID as in Pastry [9]. In others, as the Content Addressable Network (CAN) [10], a region of the space is used to map a key. Some efforts have been made to apply this solution to sensor networks [11]. When coordinates are available, sensor networks use Geographic Hash Tables (GHT) instead of a typical DHT. Currently, technology companies as Ericsson Research, are making an effort to develop applications that use GHTs in wireless sensor networks [12].

B. *Distributed and Unstructured Event-based Systems*

The main characteristics of distributed and unstructured event-based systems is that they do not maintain an overlay layer. This fact makes easier to deal with network changes. The distributed notification service may be built using flooding, gossiping or random walks.

Most of the algorithms proposed deal with the unstructured nature of wireless communications using flooding to build a tree. A typical solution is to use the On-Demand Multicast Routing Protocol (ODMRP) [13], which is based on the forwarding group concept. Groups are constructed and maintained periodically when a multicast source has data to send. This task is done by broadcasting the entire network with membership information. An extension for ODMRP has been proposed [14] to adapt a content-based system by adding subscriptions to Bloom filters. Trees also may be configured to self-repair themselves in base to brokers dynamicity [15]. These solutions are reliable but increase the traffic of the network because they use flooding at some point.

Flooding may also be used to continuously exchange subscription information clusterizing the network [16]. Then, notifications are sent to the appropriate cluster, improving the efficiency of the network. Other mechanisms can be used as the combination of a DHT and random walks [17]. Cluster heads manage the DHTs while random walks help to connect the different cluster heads of the network. The cluster concept in the network of brokers can be improved in a dynamic scenario by enriching the topology management with predictions based on location [18].

1) Probabilistic approaches: Probabilistic approaches are suitable to deal with dynamicity but they do not offer reliability. Some solutions propose that all the nodes in the network implement a broker that forwards messages to neighbors depending on the estimation of potential subscribers [19]. Other solutions [20], propose to flood subscriptions in a small area and then use random walks to reach these areas. In Quasar [21], subscriptions of a certain area are able to attract or reject notifications, that are propagated with a random walk, using an attenuated Bloom filter [22]. A probabilistic solution that uses a random walk specifically designed to go deep into the network is CoQUOS (Continous Queries on Unstructured OverlayS) [23]. Continuous queries are launched to the network using random walks. Peers compute the overlap between their neighbor lists and use this information to forward the random walk to avoid remaining in a cluster. Then, some peers register the query with a probability that depends on the number of hops.

III. Network Architecture

Due to the unstructured nature of our network, we propose the development of a dissemination algorithm that merges the network and the overlay layers of a typical distributed and structured event-based system (see Figure 2). This means that no other network protocol is needed. The main advantage of not using another network protocol is that there is no necessity of maintaining a network topology. This implies that most nodes of the network, which do not actively participate in the process of dissemination, do not have to keep any information about topology. The main consequence is that nodes not involved in the system are able to save energy and computing resources.

Our design (see Figure 3) uses two layers on the top of a bottom layer that provides data link functionalities:

1) The overlay layer is in charge of providing the distributed network of brokers and, at the same time, provides point-to-point communication between publishers and subscribers. The main objective of this strategy is to avoid the use of global information of the network, which is costly to get and maintain.
2) As in Figure 2, the event-based protocol is implemented at the top layer.

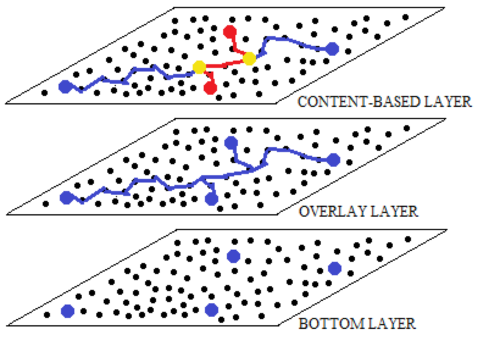


Figure 3: Decomposition of the architecture of our design in layers.

As Section I mentions, we assume that a node can be a publisher, a subscriber, a broker or a combination of these three possibilities. Moreover, our design takes advantage of some nodes in the network that want to collaborate. Nodes that participate in the system are considered as part of the overlay layer (blue path of Figure 3). The overlay layer is formed by the intersection of different publishers and subscribers (blue nodes). Publishers and subscribers implement a DRW until intersecting other DRW. Broker nodes (yellow nodes) are the meeting point between two DRWs.

A DRW is a probabilistic technique able to go forward into the network following a loop-free path. The principle assumed in this strategy is that two lines in a plane cross (see Figure 4). It is unclear how to construct a straight path of relaying nodes in ubiquitous unstructured networks without requiring global information and without making use of geo-coordinates. In this research, two different methods have been proposed in order to build DRWs [24][25].

The strategy followed by the DRW is based on a tabu search [26]. The tabu search is a technique used when difficult optimization problems arise. Unfortunately, the theoretical aspects related to a tabu search are so complicated that there is no formal proof of the convergence of the algorithm.

By definition, a tabu search is an iterative procedure in which the next solution is defined by the current solution and a tabu list. A tabu list is a memory that keeps information about the previous iterations of the algorithm. It is used to select the optimal solution for the next iteration. In neighborhood search methods, the tabu list is referred to the set of neighbors of the actual solution. The design of a DRW uses a technique, which is similar to a tabu list based on a neighborhood search method. A DRW marks the closest nodes of nodes that are already part of the DRW. Afterwards, this information is used to go forward when adding more nodes to the DRW.

The general algorithm for a tabu search based on a neighborhood method is the following:

```
Step 1   Set initial solution $S_t$ where $t = 0$. Add $S_t$ to
         the tabu list.
Step 2   Update the current number of iteration $t = t+1$.
Step 3   Create the solution neighborhood $N(S_t)$ dis-
         carding nodes that are part of the tabu list.
Step 4   If $N(S_t) = \emptyset$ then consider $S_{t-1}$ and return to
         Step 3.
Step 5   If $N(S_t) \neq \emptyset$ then evaluate the cost function
         for all $N(S_t)$.
Step 6   Select the best solution $S_{t+1}$ basing the choice
         on the minimum cost. Add $S_{t+1}$ to the tabu list.
Step 7   Stop the algorithm if the stopping criterion is
         satisfied. Otherwise, return to Step 2.
```

A tabu list needs a stopping condition. In our design, the condition is referred to an intersection with a node that is already part of another DRW in the network.

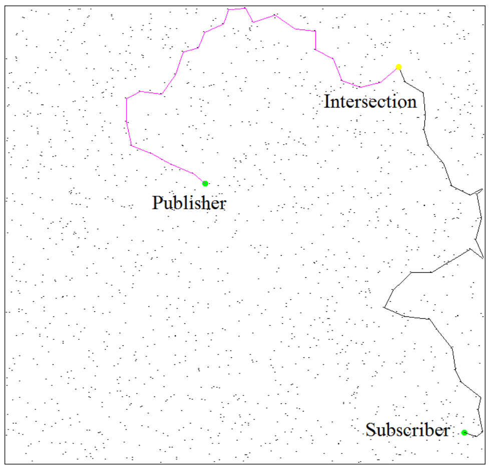


Figure 4: Directional random walks intersecting using a Java simulator.

The matching of publishers and subscribers will be done using a special architecture of Bloom filters [22] implemented at broker nodes. It is remarkable to mention that in our event-based system no advertisement table is required because filters only manage information about subscriptions.

Bloom filters are probabilistic data structures that efficiently manage membership of a certain number of elements. The content related to membership is hashed using different hashing algorithms. Then, the positions of the Bloom structure corresponding to the hashes are set to one. The maximum number of elements to be inserted to the filter is fixed in order to maintain a certain probability of false positives. When searching for elements of a certain membership, the corresponding positions of the data structure are checked. The main advantage of Bloom filters is that they do not require much memory space and processing resources; so its use is very convenient in a sensing scenario in which devices have limited capabilities.

In this research, we concentrate on the study of the properties of the overlay layer proposed. It is out of the scope of this work to study a more efficient architecture of Bloom filters at broker nodes for matching publications with subscriptions.

IV. Background

In this section, we present the efforts already made in order to build DRWs.

Based on [24], a first method to build DRWs is proposed. It is based on the addition of different nodes to the DRW by pre-computing different weights at each node that take into account the two hops path. A weight is defined as follows:

$$n^y_{xz} = \mid N(x) \cap N(z) \mid \tag{1}$$

where y is the last node added to the DRW; x is the penultimate node added to the DRW; z can be any node of the set $N(y)$ and $N(a)$ is the set of neighbor nodes of node a. Furthermore, in this method, a penalty is added to the weight when a node is added to the DRW.

Some properties about our heuristics were found using extensive simulations. The first property claims that DRWs decrease the time to intersection compared to pure random walks. The second property states that cooperation also decreases the time to intersection. Cooperation refers to synchronicity between publishers and subscribers. Finally, it is shown that DRWs are good at balancing the load of the network.

Based on [25], a second method to build DRWs is proposed. The main difference with the first design presented for DRWs is that nodes of the first and second neighborhoods of nodes added to the DRW are marked. In addition to this, the cost is not pre-computed, but it is computed when selecting a node as follows:

$$c(v) = \alpha|N(v) \cap N(DRW)| + \beta|N(v) \cap N^2(DRW)| \tag{2}$$

where α and β are parameters used as weights; v can be any node of the set related to the neighborhood of the last node added to the DRW; DRW is the set of nodes that are part of the DRW; $N(a)$ is the set of neighbor nodes of node a and $N^2(DRW)$ is the set of neighbor nodes of $N(DRW)$.

In the first part of this research, the properties associated with a DRW were assessed. Implementations of DRWs of one or two branches were studied. The main results show that the use of one branch is as efficient as the use of two branches. Moreover, it is shown that the use of second neighborhoods to forward the DRW does not improve the Euclidean distance traversed in the network. It is also shown that shorter paths are obtained when using higher densities of nodes in the network. In the second part of this research, an information brokerage system was evaluated using a double ruling method. As in the first paper, it is shown that the algorithm is efficient at balancing the load using a few nodes of the network. In fact, we can state that the method proposed is as good as a traditional Rumor Routing algorithm [27] with an infinite memory.

V. Design of the overlay layer

A. Network Model

A DRW is defined in a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. $u, v \in V$ are connected $u \sim v$ if $(u, v) \in E$. The size of G is denoted by $\mid V \mid= n$ and the number of edges is denoted by $\mid E \mid= m$. We denote the neighborhood of v as $N(v) = \{u \in V \mid u \sim v\}$.

B. Implementation of the overlay layer

In order to assess the architecture proposed, we have used a variation of the algorithm presented at [25].

The set of edges and vertices associated to a DRW of ID x are denoted by E'_x and V'_x. Our technique consists of selecting the set of vertices V'_x that are part of the DRW. Each vertex of V'_x is denoted by $v'_{x,i}$. The current number of nodes in the active path of the DRW is denoted by i. Vertices are chosen consecutively until two DRWs intersect.

A vertex v is selected to be part of the DRW as $v'_{x,i}$ if it has the minimum cost at iteration i between $N(v'_{x,i-1})$. The cost function used is a varitation of the cost function used in (2):

$$c(v) = |N(v) \cap N(DRW_{x,i})| \tag{3}$$

where $N(DRW_{x,i})$ denotes the set of neighbors of V'_x at iteration i. Formally, it is defined as:

$$N(DRW_{x,i}) = \left[\bigcup_{j=0}^{i} N(v'_{x,j})\right] \tag{4}$$

The use of $N(DRW_{x,i})$ is of particular interest to our research because it allows us to exploit the broadcast advantage of the wireless medium. This process can be seen as a repulsion mechanism to force a branch to keep moving forward. The result of this mechanism is that neighbors that are not part of $N(DRW_{x,i})$ have higher possibilities to be added to the DRW.

Figure 5 illustrates the selection of a node z when nodes x and y have already been added to a DRW. Before adding z to the DRW (see Figure 5.a), neighbors of the penultimate node added to the DRW are marked as part of the neighborhood of the walker. At this stage, node y has to select the next node to be added to the DRW between its neighbors a, b, c, d and z (see Figure 5.b). In order to avoid remaining in the same zone, we are interested in selecting a candidate that helps to push the DRW to other zones of the network. Candidate a has a cost of 3 because it has three neighbors marked as part of the neighborhood of the walker (see Figure 5.b). Candidates b, c, d, and z; have a cost of 2, 3, 2 and 1, respectively. Candidate z has the minimum cost, so that it is added to the DRW and neighbors of node y are marked as part of the neighborhood of the walker.

Publishers and subscribers are considered as initiators of a DRW. Algorithm 1 shows how DRWs are intersected for a certain number of publishers and subscribers in the network.

Algorithm 2 shows the pseudocode used for the construction of a DRW. The selection of a node is based on the computation of a cost (line 26). A candidate node is added to the DRW if it has the minimum cost between all the candidate nodes (line 30). A node is considered as candidate, if it is part of the neighborhood of the last node added to the branch (line 21). It is assumed that there are no candidate nodes when the neighborhood of the last node added is empty or all of them are already part of the DRW (line 18). In order to assure intersections our algorithm goes back in the DRW to search for the nearest non traversed neighbor.

VI. Evaluation of the novel architecture

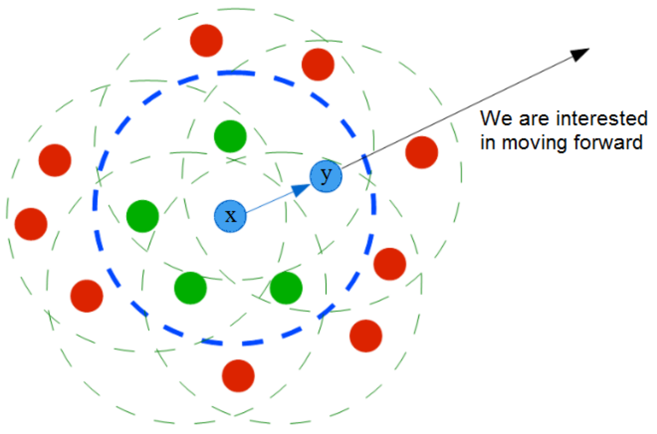
To assess the performance of the overlay layer we have implemented a Java simulator. The networks used for the numerical evaluation have been obtained by placing nodes randomly and uniformly in a squared area of side size 1×1 with a range of communication of $r = 0.05$. The communication model is defined by the range of communication. Two

Algorithm 1 Construction of the overlay layer

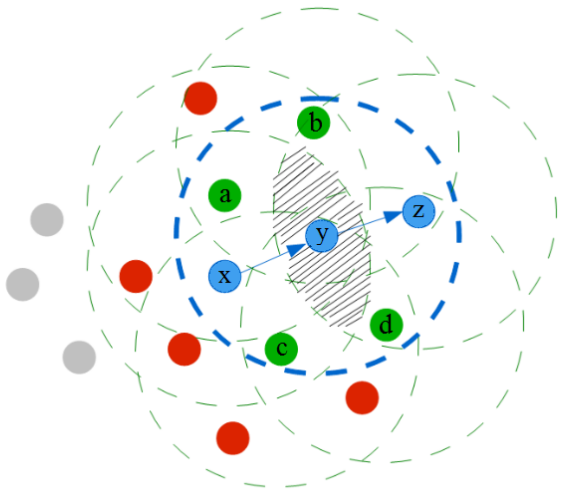
```
Require: $Number\ of\ total\ initiators : I$
1: Define Thread $drw0$ as a new DRW($v'_{0,1}$)
2: Define Thread $drw1$ as a new DRW($v'_{1,1}$)
3: Start Thread $drw0$
4: Start Thread $drw1$
5: while{(Intersection value of $drw0$ is 0) && (Intersection
   value of $drw1$ is 0)}
6: end while
7: for $i = 2; i < I; i++$ do
8:     Define Thread $drwi$ as a new DRW($v'_{i,1}$)
9:     Start Thread $drwi$
10:    while{(Intersection value of $drwi$ is 0)}
11:    end while
12: end for
```

Algorithm 2 Thread of the construction of a DRW

```
Require: $Initiator : v'_{x,1} \in V$
1: function DRW($v'_{x,1}$)
2: $Intersection = 0$
3: add $v'_{x,1}$ to $V'_x$
4: if any $v \in N(v'_{x,1})$ is part of other DRW then
5:     return $Intersection = 1$
6: else
7:     if any $v \in N(v'_{x,1})$ is part of other DRW then
8:         $v'_{x,2} \leftarrow v$
9:         return $Intersection = 1$
10:    else
11:        select $v'_{x,2} \in N(v'_{x,1})$ randomly
12:        $i = 2;$
13:        while $Intersection = 0$ do
14:            for each $v \in N(v'_{x,i-1})$ do
15:                add $v$ to $N(v'_{x,i})$
16:            end for
17:            $i++;$
18:            while $(\{v \mid v \in N(v'_{x,i})\} = \emptyset) \mid\mid$
                 $(\{v \mid v \in N(v'_{x,i})\} \in DRW)$ do
19:              $i--;$
20:            end while
21:            for each $v \in N(v'_{x,i})$ do
22:                if $v$ is part of other DRW then
23:                    $v'_{x,i} \leftarrow v$
24:                    return $Intersection = 1$
25:                else
26:                    compute $c(v)$ as defined by (3)
27:                end if
28:            end for
29:            if $Intersection = 0$ then
30:                add to the DRW $v \mid v, min\{c(v)\} \in N(v'_{x,i})$
31:            end if
32:        end while
33:        return $Intersection = 1$
34:    end if
35: end if
36: end function
```



a) Two nodes added to the directional random walk



b) Three nodes added to the directional random walk

Figure 5: Construction of a directional random walk.

nodes that are closer than the range of communication can communicate. The graph we obtain in this way is often referred by Unit Disc Graph (UDG). Under these conditions, it is hard to obtain connected networks with less than 1.000 nodes, so we have conducted numerical validation for more dense networks assuring that they are completely connected. The total number of nodes considered has been 1.000, 2.000 and 3.000.

As previously mentioned, for the implementation of the overlay layer we use different DRWs that intersect. Figure 6 shows a simulation of the overlay layer in which yellow squares represent the distributed network of brokers while publishers and subscribers are represented using green circles.

The performance metric used is the *depth* (see Figure 7 and (5)). The *depth* compares the maximum Euclidean distance reached by all the nodes that are part of the list of relaying nodes of any DRW, with the maximum Euclidean distance that can be reached in the network. It is defined as:

$$depth(Overlay\ Layer) = \frac{max\{\{d(v'_i, v'_j) \mid v'_i, v'_j \in \bigcup_x V'_x\}}{max\{d(v_i, v_j) \mid v_i, v_j \in V\}} \quad (5)$$

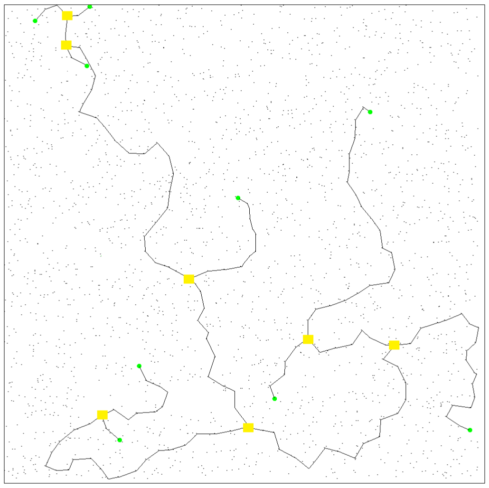


Figure 6: Overlay layer using directional random walks in a Java simulator.

where d denotes the Euclidean distance.

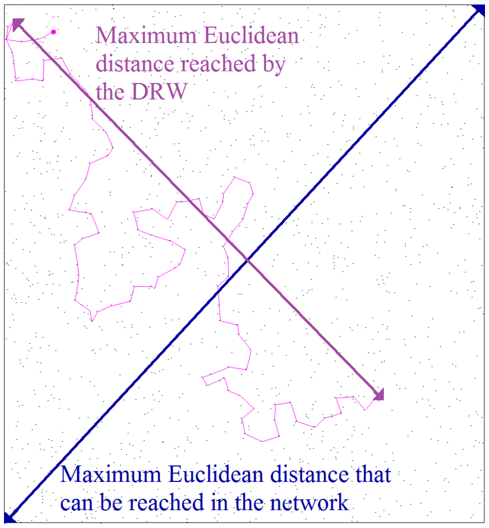

Maximum Euclidean distance reached by the DRW
Maximum Euclidean distance that can be reached in the network

Figure 7: Representation of the *depth*.

The evaluation of the design of the overlay layer is based on:

- The number of nodes in the active path.
- The *depth*.

For this purpose, we assess the performance of the architecture proposed for different number of publishers and subscribers in the network. Our results, have been obtained by using extensive simulations for each network considered. Box and whisker plots are used to visualize data. As previously mentioned, we considered networks of 1.000, 2.000 and 3.000 nodes. The total number of networks simulated totals 126.000.

The total number of simulations when evaluating networks of 1.000 nodes has been 20.000. We have evaluated the performance of the overlay layer for the following total number of publishers and subscribers: 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 75, 100, 250, 500, 625, 750 and 875. For each total number of initiators, we have simulated 100 different networks.

Similarly, for networks of 2.000 nodes we have obtained 21.000 simulations using a total number of initiators of: 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 75, 100, 250, 500, 1.000, 1.250, 1.500 and 1.750.

Finally, for networks of 3.000 nodes 22.000 simulations have been conducted for a total number of initiators of: 2, 3, 4, 5, 6, 7, 8, 9, 10, 20, 30, 40, 50, 75, 100, 250, 500, 1.000, 1.500, 1.875, 2.250 and 2.625.

Moreover, we have used PRWs for comparison with our method, which select the next node of the walker randomly. So that overlay layers formed by PRW have also been evaluated for 1.000, 2.000 and 3.000 nodes.

A. *Impact on the number of nodes in the active path*

Figure 8 compares the results obtained by using DRWs and PRWs for networks of 1.000, 2.000 and 3.000 nodes.

Interestingly, Figures 8.a, 8.c and 8.e, that show the performance of the overlay layer when using DRWs are proportionally, almost identical for different densities of nodes in the network following the same logarithmic behavior. This behavior suggests that the base of the logarithm decreases as the density of the network is increased. Nevertheless, Figures 8.b, 8.d and 8.f, that show the performance of the overlay layer when using PRWs, present a different logarithmic behavior. We observe that proportionally, less dense networks grow faster in terms of nodes in the active path; but still, the more nodes we have in the network, the more nodes we have in the active path. This means that as in the previous case, the base of the logarithm decreases as the density of the network is increased; but the change on the base is more dramatic for DRWs. This effect is produced because DRWs reduce the random component attached to the experiment, allowing to have a more predictable performance, that is traduced in more directionality or similarity to a straight line of the walker.

Besides this, overlay layers constructed using PRWs present more outliers that overlay layers that use DRWs. The result is that DRWs present a more reliable performance in the construction of the active path of our architecture. In addition to this, it is obvious, that overlay layers that use PRWs present larger active paths.

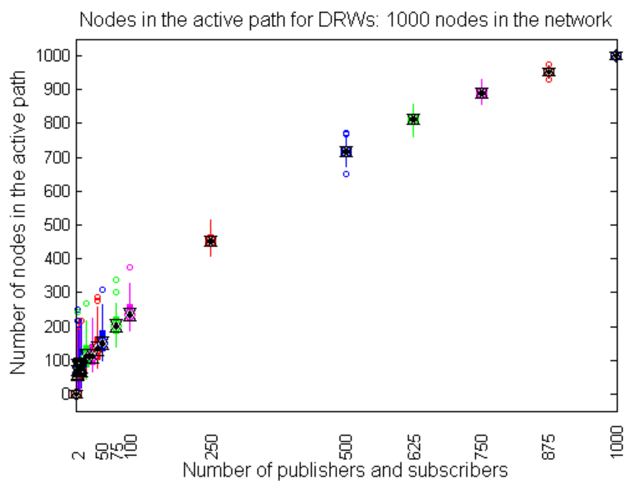
As previously mentioned, in all cases the more nodes we have in the network, the more nodes we have in the active path of the overlay layer for the same number of publishers and subscribers in the network. This consequence is reasonable, because the less density of nodes we have in the network, the less candidate nodes we have to construct the walker. The result of this performance is that less dense networks are saturated before.

B. *Impact on the depth*

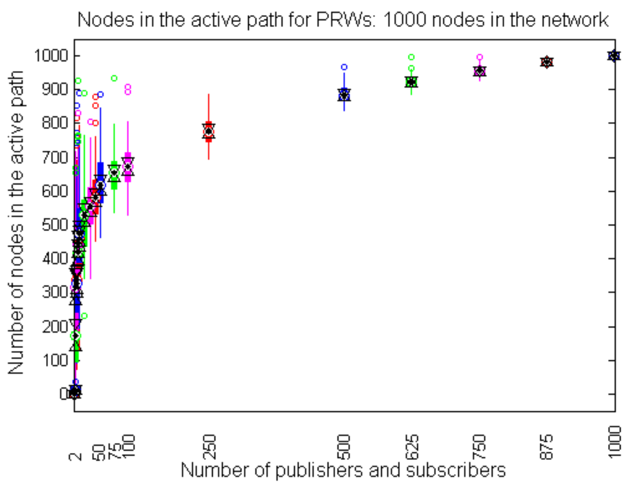
Figure 9 shows the resulting *depth* for the different densities of networks considered using DRWs and PRWs. The main conclusion extracted is that *depths* are very similar for all cases and that the maximum Euclidean distance that is going to be traversed in the network is reached very early.

Figure 10 shows in detail the performance when having a few number of publishers and subscribers in the network. In all cases, the depth is importantly increased when having three publishers and subscribers in the network. Furthermore, we observe that the directionality of DRWs leads to increase the *depth* when having two publishers or subscribers compared to the *depth* reached by PRWs.

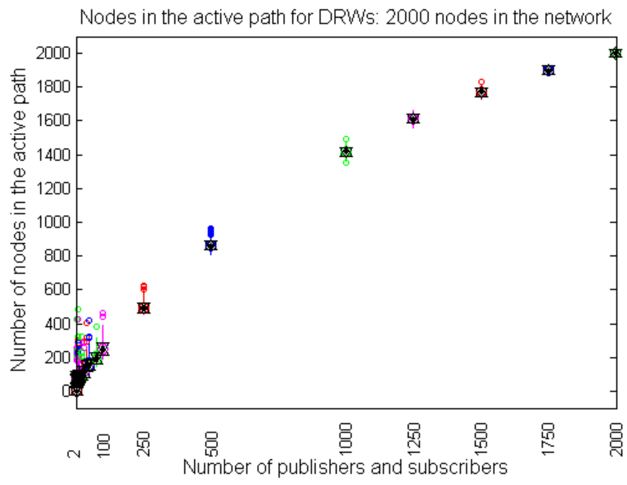
Finally, we can state that there exists no correlation between the number of nodes in the active path and the *depth*. So that other factors, as the density of the network, have more impact in the number of nodes of the active path.



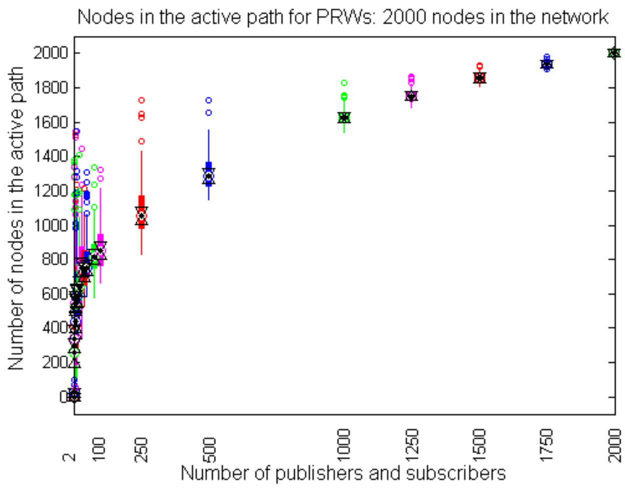
a)



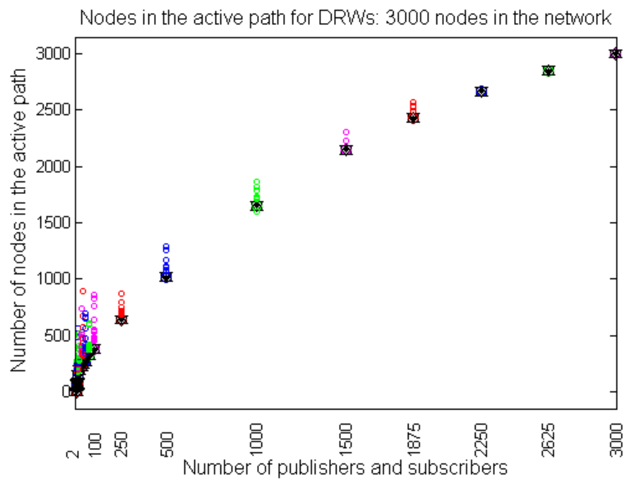
b)



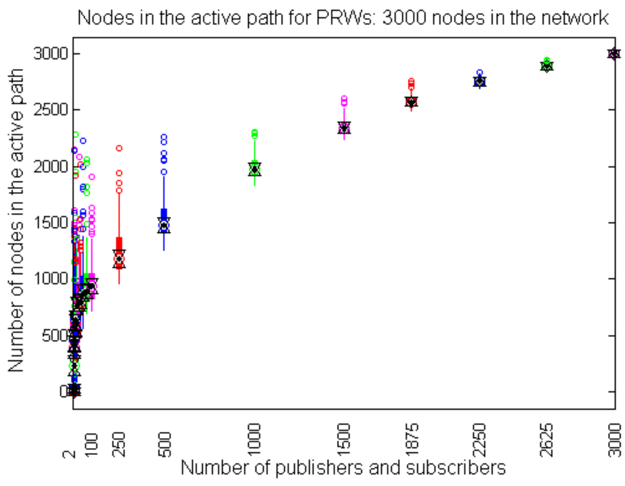
c)



d)

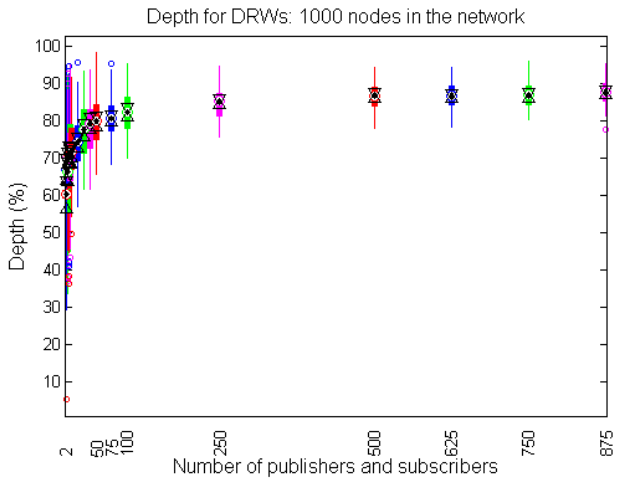


e)

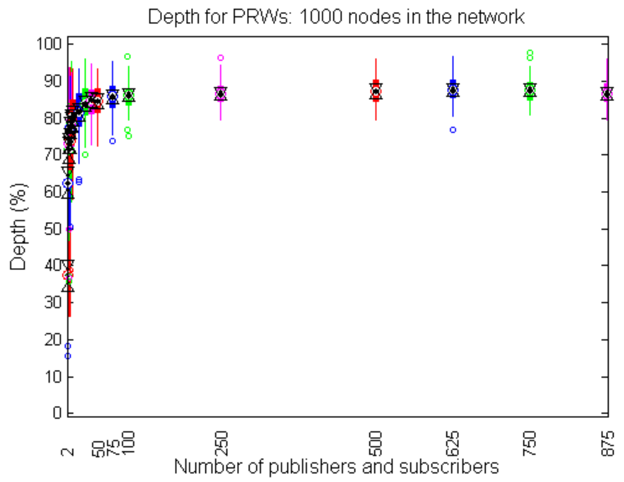


f)

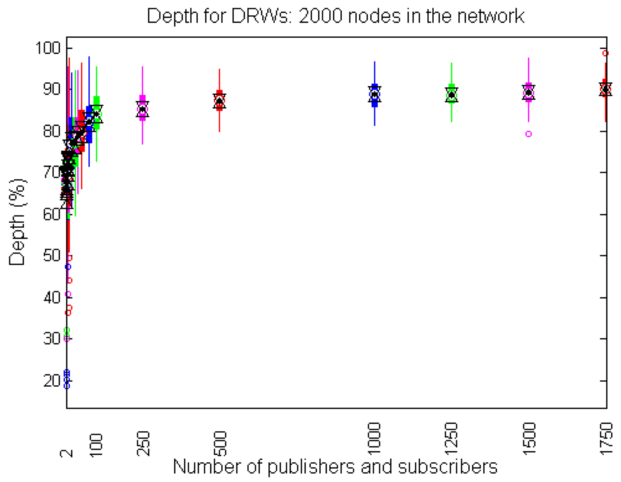
Figure 8: Nodes in the active path of the overlay layer for different number of publishers/subscribers in the network using directional random walks (a, c, e) and pure random walks (b, d, f). The total number of nodes in the network is 1.000 nodes (a, b), 2.000 nodes (c, d) and 3.000 nodes (e,f).



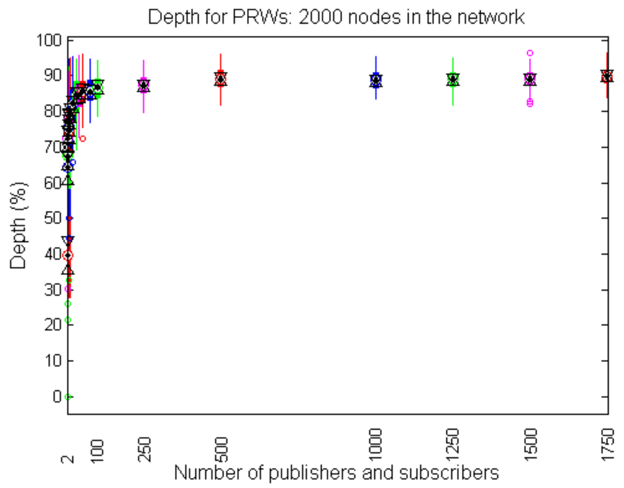
a)



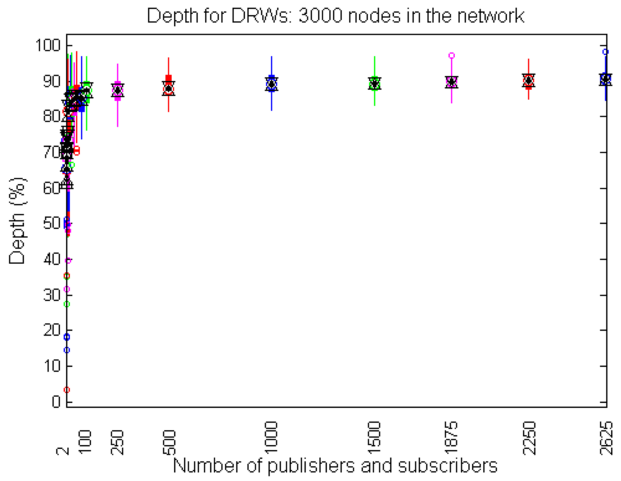
b)



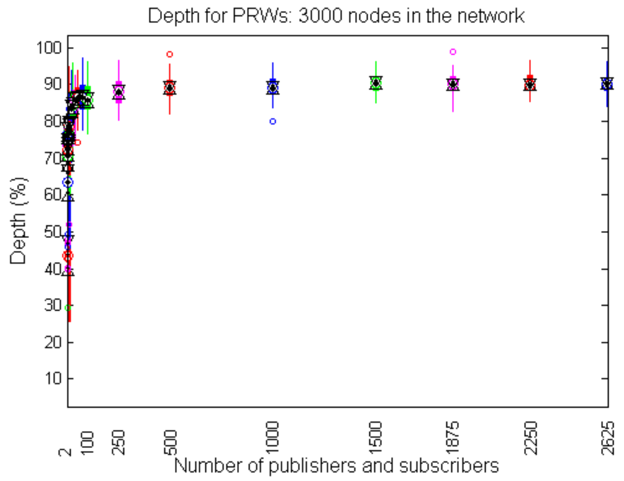
c)



d)

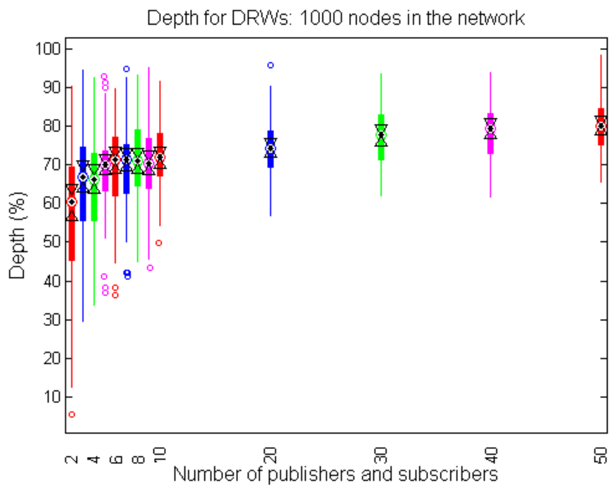


e)

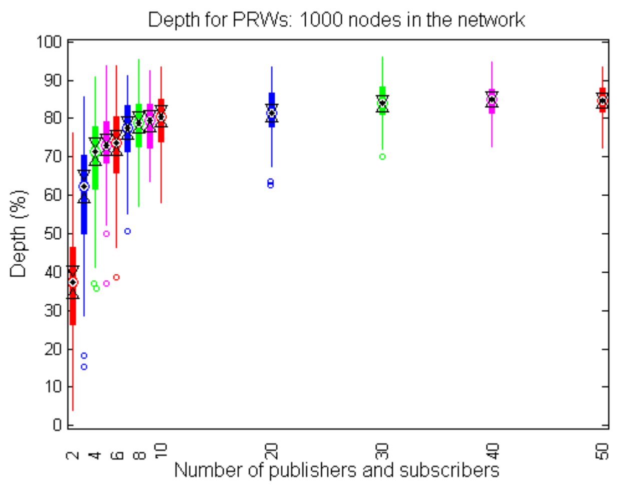


f)

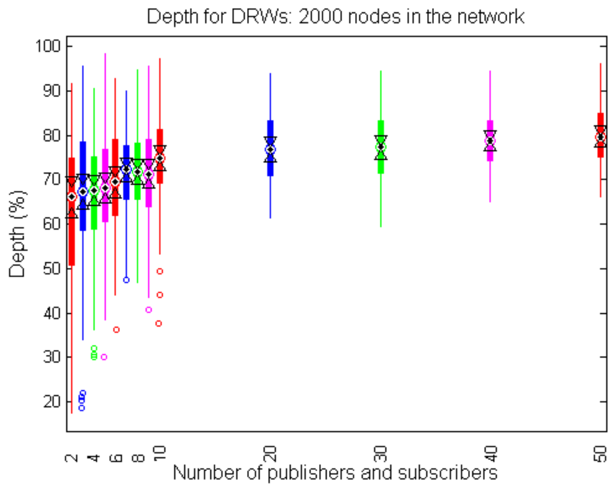
Figure 9: Depth of the overlay layer for different number of publishers/subscribers in the network using directional random walks (a, c, e) and pure random walks (b, d, f). The total number of nodes in the network is 1.000 nodes (a, b), 2.000 nodes (c, d) and 3.000 nodes (e,f).



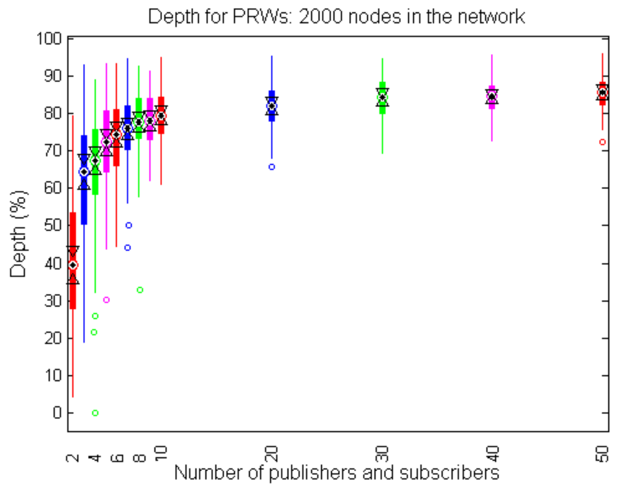
a)



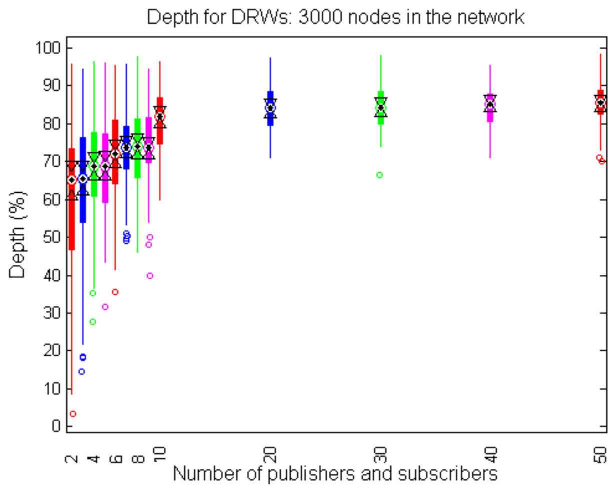
b)



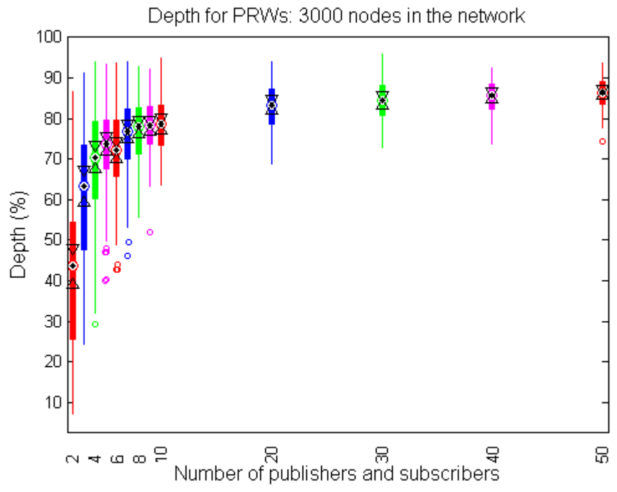
c)



d)



e)



f)

Figure 10: Depth of the overlay layer, for the first nodes in the active path, for different number of publishers/subscribers in the network using directional random walks (a, c, e) and pure random walks (b, d, f). The total number of nodes in the network is 1.000 nodes (a, b), 2.000 nodes (c, d) and 3.000 nodes (e,f).

VII. Conclusion

In this paper, a novel network architecture for distributed event-based systems that use sensing devices has been proposed. We present a network architecture that merges the network and overlay layers of typical structured event-based systems. Our results, validated through extensive simulations, show that DRWs are suitable for the construction of an overlay layer that provides point-to-point communication and a distributed notification service.

Our strategy avoids using other network protocol to provide point-to-point communication. This implies that most nodes of the network, which do not actively participate in the process of dissemination, do not have to maintain any information about topology. The main consequence is that nodes not involved in the system are able to save energy and computing resources.

We evaluate the performance of the overlay layer using DRWs and PRWs for its construction. Our results show that for our purpose DRWs are more efficient than PRWs. This is due, mainly to the good properties of DRWs, which use less nodes of the network for the establishment of the active path of the overlay layer. Moreover, we can state that overlay layers that use DRWs have a more reliable performance that overlay layers that use PRWs. Furthermore, it is remarkable to mention that, in all cases, the more nodes we have in the network, the more nodes we have in the active path of the overlay layer for the same number of publishers and subscribers in the network. Finally, it is interesting to discard any correlation between the number of nodes that form the overlay layer and the maximum Euclidean distance that is traversed by the walkers; mainly because the maximum $depth$ is quickly reached.

Acknowledgment

This work has been developed as part of the POPWiN project that is financially supported by the Swiss Hasler Foundation in its SmartWorld - Information and Communication Technology for a Better World 2020 program.

References

[1] C. Muñoz and P. Leone, “A network architecture for distributed event-based systems in an ubiquitous sensing scenario,” in Proc. 6th Intl. Conference on Future Computational Technologies and Applications, FUTURECOMPUTING, 2014, pp. 65–68.

[2] D. J. Cook and S. K. Das, “Review: Pervasive computing at scale: Transforming the state of the art,” Pervasive Mob. Comput., vol. 8, no. 1, Feb. 2012, pp. 22–35.

[3] G. Mühl, L. Fiege, and P. Pietzuch, ”Distributed Event-Based Systems”. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[4] S. Roy, V. Addada, S. Setia, and S. Jajodia, “Securing maodv: attacks and countermeasures,” in Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON. Second Annual IEEE Communications Society Conference on, Sept 2005, pp. 521–532.

[5] K. Shi, Z. Deng, and X. Qin, “Tinymq: A content-based publish/subscribe middleware for wireless sensor networks,” in SENSORCOMM 2011, The Fifth International Conference on Sensor Technologies and Applications, 2011, pp. 12–17.

[6] E. Souto et al., “Mires: a publish/subscribe middleware for sensor networks,” Personal and Ubiquitous Computing, vol. 10, no. 1, 2006, pp. 37–44.

[7] Q. Fang, J. Gao, L. J. Guibas, V. Silva, and L. Zhang, “Glider: Gradient landmark-based distributed routing for sensor networks,” in in Proc. of the 24th Conference of the IEEE Communication Society INFOCOM, 2005, pp. 339–350.

[8] Q. Fang, J. Gao, and L. J. Guibas, “Landmark-based information storage and retrieval in sensor networks,” in In The 25th Conference of the IEEE Communication Society INFOCOM, 2006, pp. 1–12.

[9] A. I. T. Rowstron and P. Druschel, “Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,” in Proceedings of the IFIP/ACM International Conference on Distributed Systems Platforms Heidelberg, ser. Middleware ’01. London, UK, UK: Springer-Verlag, 2001, pp. 329–350.

[10] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A scalable content-addressable network,” in Proceedings of the 2001 conference on Applications, technologies, architectures, and protocols for computer communications, ser. SIGCOMM. New York, NY, USA: ACM, 2001, pp. 161–172.

[11] G. Fersi, W. Louati, and M. Ben Jemaa, “Distributed hash table-based routing and data management in wireless sensor networks: a survey,” Wirel. Netw., vol. 19, no. 2, Feb. 2013, pp. 219–236.

[12] H. Mahkonen, T. Jokikyyny, J. Jiménez, and S. Kukliński, “M3: Machine-to-machine management framework,” in Proc. 3rd Intl. Conference on Sensor Networks, SENSORNETS, 2014, pp. 139–144.

[13] S. J. Lee, W. Su, and M. Gerla, “On-demand multicast routing protocol in multihop wireless mobile networks,” Mob. Netw. Appl., vol. 7, no. 6, Dec. 2002, pp. 441–453.

[14] E. Yoneki and J. Bacon, “An adaptive approach to content-based subscription in mobile ad hoc networks,” in Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, ser. PERCOMW. Washington, DC, USA: IEEE Computer Society, 2004, pp. 92–97.

[15] L. Mottola, G. Cugola, and G. P. Picco, “A self-repairing tree topology enabling content-based routing in mobile ad hoc networks,” IEEE Transactions on Mobile Computing, 2008, pp. 946–960.

[16] S. Voulgaris, E. Rivire, A.-M. Kermarrec, and M. V. Steen, “Sub-2-sub: Self-organizing content-based publish subscribe for dynamic large scale collaborative networks,” in In the fifth International Workshop on Peer-to-Peer Systems, IPTPS, 2006.

[17] R. Tian et al., “Hybrid overlay structure based on random walks,” in Proceedings of the 4th international conference on Peer-to-Peer Systems, ser. IPTPS. Berlin, Heidelberg: Springer-Verlag, 2005, pp. 152–162.

[18] F. Abdennadher and M. Ben Jemaa, “Accurate prediction of mobility into publish/subscribe,” in Proceedings of the 11th ACM International Symposium on Mobility Management and Wireless Access, ser. MOBIWAC. ACM, 2013, pp. 101–106.

[19] J. Haillot and F. Guidec, “Content-based communication in disconnected mobile ad hoc networks,” in Proceedings of the 8th international conference on New technologies in distributed systems, ser. NOTERE. New York, NY, USA: ACM, 2008, pp. 21:1–21:12.

[20] P. Costa and G. Picco, “Semi-probabilistic content-based publish-subscribe,” in Distributed Computing Systems, ICDCS. Proceedings. 25th IEEE International Conference on, 2005, pp. 575–585.

[21] B. Wong and S. Guha, “Quasar: a probabilistic publish-subscribe system for social networks,” in Proceedings of the 7th international conference on Peer-to-peer systems, ser. IPTPS. Berkeley, CA, USA: USENIX Association, 2008, pp. 2–2.

[22] S. Tarkoma, C. Rothenberg, and E. Lagerspetz, “Theory and practice of bloom filters for distributed systems,” Communications Surveys Tutorials, IEEE, vol. 14, no. 1, First 2012, pp. 131–155.

[23] L. Ramaswamy and J. Chen, “The coquos approach to continuous queries in unstructured overlays,” IEEE Trans. on Knowl. and Data Eng., vol. 23, no. 3, Mar. 2011, pp. 463–478.

[24] P. Leone and C. Muñoz, “Content based routing with directional random walk for failure tolerance and detection in cooperative large scale wireless networks,” in Proc. 2nd Intl. Workshop on Architecting Safety in Collaborative Mobile Systems, SAFECOMP, 2013, pp. 313–324.

[25] C. Muñoz and P. Leone, “Design of an unstructured and free geo-coordinates information brokerage system for sensor networks using directional random walks,” in Proc. 3rd Intl. Conference on Sensor Networks, SENSORNETS, 2014, pp. 205–212.

[26] M. Gendreau and J.-Y. Potvin, “Tabu search,” in Search Methodologies. Springer, 2014, pp. 243–263.

[27] D. Braginsky and D. Estrin, “Rumor routing algorthim for sensor networks,” in Proceedings of the 1st ACM international workshop on Wireless sensor networks and applications, ser. WSNA. New York, NY, USA: ACM, 2002, pp. 22–31.

Moving Towards Distributed Networks of Proactive, Self-Adaptive and Context-Aware Systems: a New Research Direction?

Remus-Alexandru Dobrican and Denis Zampunieris
Computer Science and Communication Research Unit
University of Luxembourg
Luxembourg, Grand Duchy of Luxembourg
Email: {remus.dobrican, denis.zampunieris}@uni.lu

Abstract—**Instead of being static and waiting passively for instructions, software systems are required to take a more proactive approach in their behaviour in order to anticipate and to adapt to the needs of their users. To design and develop such systems in an affordable, predictable and timely manner is a great software engineering challenge. Even though there have been notable steps for modelling self-adaptive and context-aware systems, there is still a lack of a generic model agreed by the research community for developing smart applications. The goal of this study is to explore the idea of having multiple networks of proactive context-aware adaptive systems working together for achieving common goals. To support our vision, we introduce a context-aware self-adaptive software model for mobile devices capable of learning from the user's behaviour by using Proactive Computing. The novelty comes from the possibility of developing smart applications that would benefit from the proposed properties. Moreover, we discuss a motivating scenario that led to this work and propose a case study where a collaborative e-Learning application is implementing our model.**

Keywords-smart applications, self-adaptive systems, context-aware systems, proactive computing, distributed networks.

I. INTRODUCTION

This paper is an extension of the work-in-progress presented in [1], where we introduced a vision showing the tendency of moving towards various networks of context-aware, proactive and self-adaptive systems. In this extended version, we support our vision by analyzing the latest proposed middleware, architectures and applications that try to incorporate the mentioned properties, by providing a motivating scenario and by proposing a model for software systems capable of integrating all the above properties.

The demand for devices and applications that are able to adapt their behavior at run-time, as a response to the increasing demands of users, has risen considerably in the last couple of years [2]. Giving instructions to complex software systems is becoming quite a difficult task for the users, as it requires their continuous involvement, a set of advanced technical skills and a lot of knowledge about the system. As a consequence, our model is leading the users towards new ways of interacting with smart systems that will be able to perform a variety of automated tasks on users' behalf. Three main properties are to be distinguished when speaking about systems that dynamically adapt themselves according to the context variation or the requirements change: *Self-Adaptation*, *Proactivity* and *Context-Awareness*. Modeling such systems is a difficult task because important aspects like the user's needs, the settings of the environment in which they are deployed and the all the other requirements have to be taken into account. There is no standard model agreed upon by the research community, which could serve as a common basis for developing smart applications.

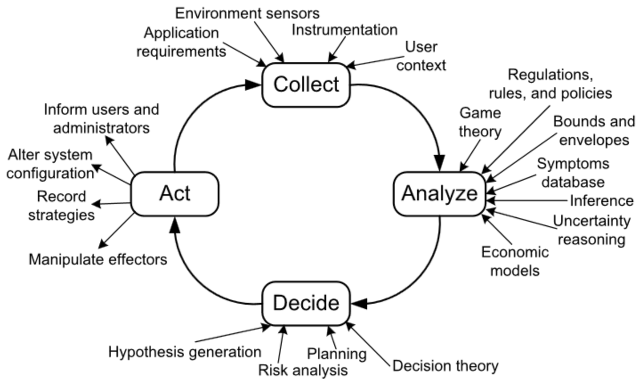


Figure 1. Autonomic control loop [4]

Self-adaptation in software systems comes in many different aspects. Self-adaptive mechanisms provide the necessary means to make changes either at an architectural level or at a behavioral level. Systems that possess this property can be characterized by their operating mode that permits them to fulfill easily their goals in a modified context.

Feedback loops provide an architectural solution for self-adaptation. In [3], the authors indicate that feedback loops usually include four key activities: *collecting*, *analyzing*, *deciding* and *acting*. These activities are essential for achieving self-adaptability, context-awareness and proactivity. In Fig. 1, a generic model of a unidirectional feedback loop is given. It shows the inputs or the outputs of each state but the data flow between the states is omitted. During the first stage of the feedback cycle, the collection of data, relevant context information can be acquired by the system from the environmental sensors and from other internal and external sources. Then, the data is analyzed by the system according to its policies and constraints, while taking in account the user's preferences. The process of transforming raw data into relevant context-information is quite problematic and complex because of many errors and broken sensors. A solution for this particular problem was proposed in [5]. Once the system reaches the third phase it will reason if future actions are needed or the system can get back in its initial state of collecting information. If future actions are required, then the system enters in the fourth state, were the actual adaptation happens. And here comes one of the biggest challenges: should the system have enough authority to perform adaptation in all cases?

The above feedback loop was inspired from IBM's four-stage cycle for autonomic computing called MAPE-K [6]. One of the main differences is that the MAPE-K model uses a knowledge base shared between all the stages.

Context-aware systems open new dimensions and opportunities for developers to create smart applications, especially for mobile devices. These systems are designed to continuously analyzing contextual information, which is a key feature for determining the occurrence or the lack of events. When a system is aware it means it acquired knowledge about the surrounding environment by its own means. Awareness refers to the state of being aware in which a system can be found. Various sensors capture the data that creates context information. They can be classified in three categories: physical, virtual and logical [7]. Physical sensors are the most frequent used type of sensors and they can provide data about location with the help of GPS, GSM, RFID tags, optical data from the cameras and IR sensors, motion data from accelerometers and gyroscopes, and ambient data collected from thermometers and barometers [8]. Virtual sensors can capture context data from applications and services including operating system events, application data and network events. Logical sensors combine data received from the other two types of sensors and they provide higher-level context information.

Events play a central role in the lifecycle of software systems. They range from simple request for different services to serious incidents that prevent the well functioning of a system. Events can be divided into three main categories: foreseen (*taken care of*), expected (*planned for*) and unexpected (*not planned for*) [9].

Tennenhouse [10] first introduced Proactive Computing as a new mode of operation that was crucial for moving towards human-supervised computing. The essential features of proactive systems, as seen in [11], are taking decision for their users and acting on their own initiative. Proactive Computing is a solution for foreseeable events, while Context-Awareness and Self-Adaptation handle unforeseen events, which are seen as deviations from normal situations.

The contribution of this paper is three-fold. First, it examines the most relevant work in several research fields like Proactive Computing, Context-Awareness and Self-Adaptation, for pointing out current research direction, what is missing and what are the next steps to be taken. Second, it offers an middleware architecture for supporting smart mobile applications, capable of performing automated tasks for the user, of analyzing large quantities of data and of making decision in different contexts. And third, it provides an analysis of a distributed network of mobile proactive systems that are implementing our model.

The rest of the paper is organized as follows: Section II provides an overview of related work relevant to our research, in Section III we describe a motivating scenario that lead to this work, Section IV describes the main components and the characteristics of a Proactive Engine, Section V investigates the possibility of having multiple distributed networks of Proactive Engines, Section VI provides an example of a smart application based on our model, Section VII proposes multiple domains where applications using our model could be developed, Section VIII contains a brief implementation overview and, in Section IX, we conclude and we emphasize the potential of Proactive Engines.

II. RELATED WORK

In this section, we focus mainly on previous work done in the area of *Proactive Systems*, *Context-Aware Systems* and *Self-Adaptive Systems*, which indicates an intensive effort to develop middleware, architectures, frameworks and prototype applications that would serve in dynamic environments. These research initiatives show the growing interest for developing a new kind of intelligent systems that will be able to perform complex tasks. Our contribution in this paper has been to find a platform that unifies multiple research efforts in the above research fields.

A. Proactive Systems

The concept and the structure of the first Proactive Engine (PE) were created in 2006 [12]. It was among the first systems specially conceived to use Proactive Computing for achieving its goals. The PE was designed as a complex mechanism for running Proactive Rules. A Proactive Rule is a structure conceived to perform specific actions in case a special situation was detected or in case of the lack of an event. The detection of students that did not submit their online assignment and the notification of their professor as a consequence, is a concrete example of a rule, which was used in a real-case scenario, when the initial Proactive Engine was deployed aside a Learning Management System (LMS) [13]. Results showed that major limitations of a LMS such as the restricted interaction and limited collaboration between learners and educators inside courses could be overcome with the help of a Proactive Computing [14].

In [15], a mechanism for creating, organizing and developing Communities of Practice (CoPs) inside a LMS using Proactive Computing was proposed. Three types of virtual CoPs were developed: for students coming from the same country, for students living in the same cities and for students enrolled in the same study program. The LMS was capable of automatically enrolling students in these communities, creating tools like forums, chat and folders for stimulating collaboration inside the CoPs and adjusting the size of the CoPs. The Proactive System performed these operations without any explicit command from the administrator of the LMS.

Previous works [15], [16], focused until now on applying Proactive Computing on a single system, thus exploring only the possibility of having only one centralized Proactive System. But a centralized solution can become quite fast non-scalable in many scenarios where a Proactive Engine handles a big number of devices and applications. Using a single Proactive System presents certain limitations and cannot benefit from the advantages that multiple Proactive Systems, working together and collaborating, would have to offer.

The possibility of having multiple networks of connected Proactive Engines, which are able to communicate, to create a shared knowledge base, to perform joint actions and to learn from each other, was not yet explored.

B. Context-Aware Systems

Even though there is an intensive effort to define a standard context-aware model for developing smart applications [17]-[20] there is a lack of precise guidelines on how to integrate context-aware mechanisms into these applications. A basic approach for modeling the adaptation requirements, SOTA, is presented in [21]. In particular, it was conceived to enable early requirements verification, the identification of knowledge requirements and the most suitable self-adaptive patterns. *Cinemappy*, a location-based application that is able to compute contextual movie recommendations [17], is a simple example how contextual information related to the temporal and spatial position of the user is exploited for obtaining personalized recommendations results for each user. Pushing relevant application to mobile devices, according to the user's location, using a context-aware architecture called *MoBe* [18] is another example for addressing cases where the location of the devices is constantly changing. *MoBe* allowed data exchange and joint actions for reaching a common goal between the cooperating devices. The design process of *OnRoute*, a context-aware mobile travel application for handling navigation in public transportation was studied in [19]. A smart way of using location information for facilitating navigation was incorporated in the application, which was able to learn from the user's behavior and propose other route alternatives. *Museum Guide,* an indoor location-based, context-aware and video on demand application [20], was able to detect if a user was moving, and, based on this contextual information, to play a certain video, related to the nearby exhibit.

Furthermore, the role of context-information for improving collaboration between mobile devices by delivering relevant information to the participants involved in the cooperation process is investigated in [22].

C. Self-Adaptive Systems

In many disciplines, ranging from Artificial Intelligence to biology, self-adaptation of software systems has been studied and only recently the software engineering community has realized the major role this property plays in developing and implementing software systems that are able to comply with important changes in the environment and in the requirements [23].

In the latest research literature many approaches study adaptive systems. However, a great amount of them are concentrated on architectural adaptation [24][25][26], parametric adaptation [27][28] or implementation adaptation [29]. More precisely, they focus on changing the internal structure of a software system, on modifying the necessary parameters of the system and on changing the implementation of different components of a software system without changing the interface. In comparison, we propose a model that allows developers to reason about adaptation at a higher level of abstraction, without having to take care of the low-level implementation details.

Multiple studies investigate how to apply adaptation mechanisms for mobile applications. For example, in [30], a battery-efficient architecture was proposed for building battery-aware applications. More recent work [31] studied a mechanism for an effective adaptive real-time multimedia content delivery for smartphones using a hybrid mobile application development platform. A mobile learning case study was discussed based on the hybrid learning application.

D. Context-Aware Adaptive Systems

Event though many researchers are treating Context-Awareness and Self-Adaptation as two separated research fields there have been numerous studies aimed to provide applications, tools and frameworks that can incorporate and provide both properties.

In [32], the authors acknowledge that embedding mechanism for achieving context-awareness and self-adaptation in pervasive systems is crucial. They propose a model-driven engineering approach able to integrate such mechanisms. A separated context model but related to the adaptive model is proposed in [33] for modeling and realizing context-aware adaptive software systems. Management context is differentiated and separated from operational context in order to obtain run-time adaptation. A tool, CAST, was recommended for modeling the system's implementation and of verifying and validating the system's context-aware adaptive behavior.

In order to address problems like the high complexity of modeling the behavior of context-specific application and of structuring application code for efficient switching between various functions of the system, a mobile middleware was discussed in [34]. This middleware system was created on top of the Android operating system as tool for building adaptive, context-aware ubiquitous mobile applications. A prototype application, i.e., a context-aware Instant Messenger, was set up using the proposed middleware and compared with previous versions of Instant Messenger. A generic adaptation model was analyzed in [35] for helping designers to develop ambient assisted living applications that enabled elderly people to live longer in their residential homes. Furthermore, a context-aware and adaptive learning scheduling framework for supporting the student's daily routines was introduced in [36].

III. MOTIVATING SCENARIO

The following scenario has motivated the need of having groups of proactive context-aware adaptive systems capable of communicating and sharing information in a transparent way. When scheduling a meeting, the users have to give many complex instructions for arranging the exact time, location and group of participants. They have to be involved in each steep of the selection process, which requires certain technical skills. Also, they have to manually check other applications like the integrated agenda or calendar to be sure there are no overlapping activities. Below, the motivating scenario is presented with the help of several scenes.

Scene 1: A professor, the head of a research department at a university, receives an email from the human resource department about the budget for the next year. As a consequence, he/she would like to arrange a meeting with his colleagues, also professors from the same research department, for discussing and who would the budget be spited to cover the expenses of each professor and his/her team of assistants for research materials, for traveling purposes and for participating at different conferences/workshops.

Scene 2: He/she then uses the smart meeting application on his/her smartphone for scheduling a meeting with the other professors. Inside the application, he/she can select different users or different target groups of users for proposing a meeting. These groups are either arranged manually or automatically, based on previous activities performed together with various users. Then, he/she would select the period when he/she would like for the meeting to take place. For instance, the time interval would be the next 2 days, starting from the moment of the proposal. At this stage, the application would start the communication process with the other smartphones for finding a possible date and time for the meeting.

Scene 3: The application starts to perform internal and external processes for finding relevant information that would allow it to find a possible solution for the meeting. Internally, the application would look for information on other applications like the integrated calendar application or the Google Calendar, in case the users have a Gmail account, or in the Outlook Calendar where the users could have additional information about future meetings. Externally, requests are sent by the application to the mobile phones and tablets of the other participants. These requests start to be processed locally by the other devices that start to look for relevant context information that would allow them to find if their users are available or busy in the interval of time proposed by the initiating device. And so, a

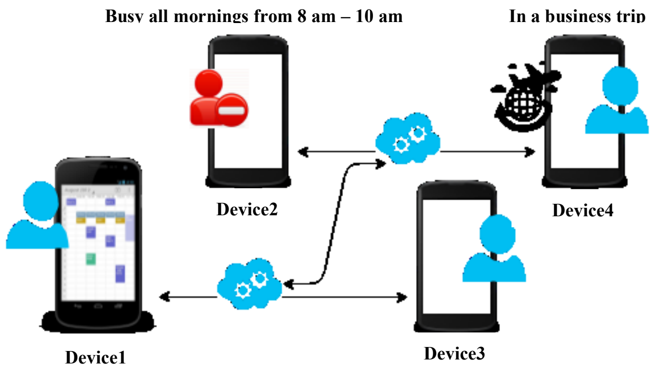


Figure 2. Establishing a meeting between smartphones

negotiation process starts between devices for finding the best solution to arrange the meeting.

Scene 4: Devices start to share information about the potential time slots that are available for the meeting as illustrated in Fig. 2. For instance, the application detects that the user of device 4 is at a remote location for the next 24 hours and cannot participate in any meeting close to the location of the other users. The user of device 2 is busy all the days of the week in the morning because he/she has to give classes from 8 am until 10 am. The two other users do not have anything scheduled in the next 2 days and are available for a meeting. After several rounds of negotiation, they find a free time slot in the next day where everyone can attend the meeting.

Scene 5: At the end of negotiation process, after an optimal solution is found and agreed upon by all the devices, a confirmation request is sent to all the users for making them aware of the meeting and to get their final approval. If there are users that do not agree with the solution then new rounds of negotiation begin until either another solution is found or the user that initiated the meeting stops the whole process.

Scene 6: If an agreement was reached about the day and the hour of the meeting, the application can start to finalize scheduling the meeting by proposing different locations. Locations can be either introduced manually by the users or can be proposed by the application based on multiple factors like previous locations or available conference rooms on the campus, information that can be found on the research department's server.

Possible issues can arise from the fact that some users could not be reached because their devices are either offline or they are turned off. In these cases, where a universal solution involving all the participants could not be found, the application either proposes a partial solution, for a meeting with the users that were found available, or the meeting is postponed until all the users will be available. In both cases, the user that initiated the meeting procedure will be notified about the outcome of the negotiation process and he/she could take additional measures, and the other participants would receive mails and/or SMS to be informed that they were requested to join a very important meeting.

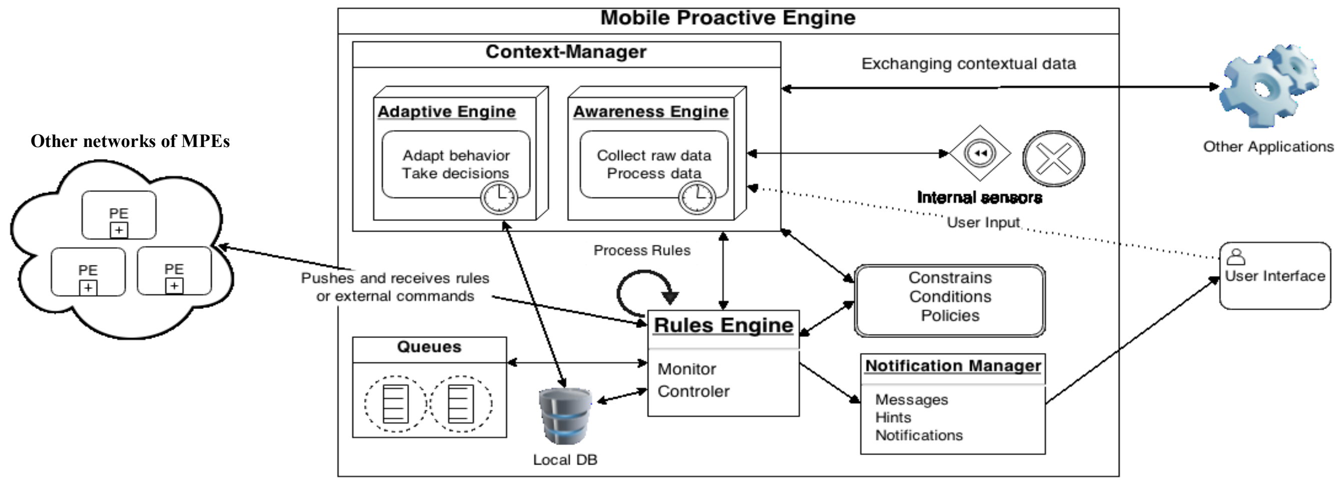


Figure 3. The structure of a Proactive Engine for mobile devices

Our goal, in this scenario, was to present an application capable of performing automated actions on behalf of the user, e.g., automatically search for possible free time slots for organizing a meeting. Therefore, the following model proposed in the next section allows the creation of such an application that implements all the above features.

IV. THE APPROACH

We propose a new version of the Proactive Engine for mobile devices, called Mobile Proactive Engine (MPE), where processes are divided between the sub-parts of the model. Before, Proactive Rules were taking care of data acquisition, activation guards, conditions, actions and rules generation. In the current model, each step is assigned to a specific component. A major benefit of separating these processes is that they are handled by structures that are focusing only on particular tasks.

In order to develop a proactive context-aware adaptive system, an infrastructure that combines and uses all three properties is required. The MPE is an advanced mechanism that could be easily integrated into new software systems because it provides means for gathering data from the internal and external sensors, for detecting context changes, for processing and modeling contextual information, for executing adaptive tasks and for providing an adequate system behavior in any situation. The term “sensor” refers not only to the hardware parts being able to sense but also to the various data sources that may give contextual information.

Thus, the architecture of a MPE is composed of a set of interconnected components, including a Context-Manager, a Rules Engine connected to a set of Queues and a local database, and a Notification Manager, as seen in Fig. 3. These components are able to communicate, sending and receiving messages or specific commands from the other components. For example, a Proactive Rule in the Rules Engine may require some additional information for responding to a situation, and would be able to activate the Awareness Engine. Then, additional information would be acquired from the sensors and, after verifying local constraints and conditions, it would be send back to the Proactive Rule that asked for additional data. The components of a MPE cannot perform structural adaptation processes. They are oriented for helping the MPE to perform a behavioral adaptation, where the initial functionalities of the MPE can be changed.

A. The Context-Manager

This component is very important, as it behaves as a filter for the majority of the data acquired by the MPE. The need of a Proactive Filter is justified by the huge amount of unnecessary data gathered by a MPE. Data coming from different sources can contain many errors or mistakes. Detecting inconsistency in the data, allows the Context-Manager to find possible broken sensors or, even more, avoid unpleasant situations where precise information is needed. The Context-Manager is mainly responsible for detecting and handling context changes that appear, and as a result, taking the proper actions. Another important task for the Context-Manager is to acquire user input and to decide if it is relevant or not. It is composed two elements: the Awareness Engine and the Adaptation Engine.

1) The Awareness Engine

It is the main component of the Proactive Filter. It is managing the data coming from sensors, which are in charge of detecting possible context changes. For smartphones, physical sensors are providing important information about the user’s location, motion and mobility. Also, other information that comes from logical and virtual sensors like the user’s interests, activities and set of used applications is constantly analyzed. Accessing this kind of information should be limited to some extend and controlled as it represents a privacy issue.

2) The Adaptation Engine

This component is crucial, as it is used for dealing with *unexpected events* and for ensuring that adaptive actions are performed in a smooth cooperation between the main sub-parts of the Proactive Engine. Also, it has to check the constraints and the conditions of the system before adaptation and if the system will still behave according to its policies. For example, adaptation could involve the interface of an application that can be modified according to the needs of the user. If a user is on the move and the application detected a speed higher than 10 km/h while the user is using the application, the interface would be change to contain a layout with bigger button, bigger writing and brighter colors.

B. *The Rules Engine*

The Rules Engine is responsible for maintaining a precise overview of the system's goals and for running Proactive Rules. It keeps a list of required actions that would come as a response in case *expected events* occur. It is also used for storing the state of the system. Executing multiple Proactive Rules in parallel is due to its integrated Queue System and it is one of the great functionalities of the Rules Engine. The Rules Engine executes Proactive Rules in *Iterations*. A constant time interval is set between two consecutive *Iteration*s. It has a default value, but it can be adjusted depending on the performance of the device that is running the MPE. For example, initially, the time interval is set to 5 seconds, as there are cases where the MPE needs to check for events in a periodic manner. If at one Iteration, the Rules Engines has to execute a big number of Proactive rules, the time interval can be modified accordingly in order to allow resources to be efficiently shared between multiple threads and processes that are running on the MPE.

Proactive Rules can be used for serving multiple purposes: for checking context situations, for detecting special events, for analyzing contextual information, for synchronizing sub-parts of the model, for saving useful data into the Local Database, for sending rules and commands to other PE and for sending content to the Notification Manager. The Awareness Engine and the Adaptation Engine also posses the ability of activating Proactive Rules as well as the other way around. Another important property of WProactive Rules is that they can run at each Iteration of the Rules Engine or, in case they perform only simple actions, they can run only once and then finish their execution.

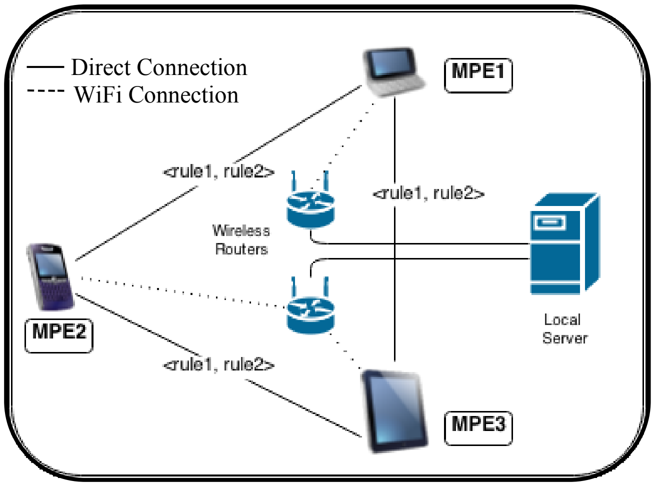
C. *The Notification Manager*

The purpose of the Notification Manager is to deliver informative content to the user. The content can take various forms like hints, messages, notifications or alarm. This is a crucial part of the entire model as it helps in achieving his/her goals, guides him/her in multiple situations and informs the user about certain events. The Notification Manager is in a close cooperation with the device's Operating System for handling messages. Proactive Rules prepare the content of the message, which is then forwarder to the user through the Notification Manager. For mobile phones, notifications can appear on the screen for short periods of time, messages can be registered by the Operating System and can be read later on by the user, hints are displayed as short text boxed for guiding the user when he/she is interacting with different applications and alarm, which can also be set by Proactive Rules, are registered in the integrated calendar and triggered for announcing the user in case of an upcoming important event.

V. NETWORKS OF MOBILE PROACTIVE ENGINES

MPEs are designed to work both offline and online. Having a network of distributed MPEs that communicate and exchange data provides a great opportunity for these systems to gain useful information. This way, MPEs are not only gathering data from their internal sensors but also from other MPEs. By design, information sharing between devices using MPEs is conceived to be done in a transparent way, without the implicit command of the user. Only in case of special situations and because of privacy issues, where sensible data is involved, the users should be asked to take over and decide what would be the next action the MPE should take. The most significant aspect to be taken into consideration is the actual information that is gained by a MPE when it gets data from other MPEs. One case is to find common interest or preferences between users that are working with applications having an integrated MPE. For example, a user could be looking for a ride on a car-sharing web site. Another user, which would be located nearby, maybe from the same city, would be looking for a ride having the same destination and exactly on the same dates. The MPEs would notify both users and would propose to share a ride for reducing the costs. Another case where data exchanging is useful is when a MPE is not sure what action to take and how to adapt its behavior when *unexpected events* happen. Requesting feedback from other MPEs that have more information is a possible solution for taking the right decision.

If we take, for example, two MPEs, one that was offline for a long period of time and one that was online during the



Local Area Network

Figure 4. A possible network of distributed MPEs

same period of time. And now, both of the MPE should be able to share information because they would be having access to a communication channel between them. The MPE that was offline could learn a lot from the online MPE that stored information about its previous tasks and about the older state of the system, without using the Adaptive Engine, the Awareness Engine and the Rules Engine to process similar data and to go through the same adaptation process. As a consequence, local resources and time could be saved.

A special aspect to taken into consideration is the interaction of the users with their MPEs. Their behavior can be analyzed by the MPEs and stored for future decisions. If a MPE would detect, for instance, that its user is changing the mode of the phone each morning on the first day of the week, due to certain activities, it could start performing this action automatically on behalf of the user. The user is important as it still remains in the processing loop, but only for supervising purposes and for providing input in cases where information cannot be acquired from different sources. Sometimes access to personal information like the location of the user, his/her local preferences, his/her status, the applications that he/she is using or his/her list of contacts should be only granted with the special consent of the user.

Fig. 4 shows a possible scenario of a network of distributed MPEs. Three devices, with a running MPE, located on the same LAN, are connected to the Internet through a WiFi connection. A direct connection can be also established via Bluetooth, via Near Field Communication (NFC) techniques or via Android's WiFiP2P library for smartphones. The advantage of having a direct connection between the devices, illustrated in Fig. 3 with a straight line, is that Proactive Rules are exchanged immediately, without having to be sent firstly to a server. This means that each device equipped with a MPE will be acting like a server, being able to receive and send data to other devices having an integrated MPE.

Different network topologies can be employed for creating groups of MPEs: centralized, hybrid and peer-to-peer. The centralized topology means that a server can be used to handle the connections and the communication of MPEs. This is the easiest way to ensure their communication but it has a major flaw, a single failure point, the server. In order to address this issue, a hybrid topology can be used, where multiple servers are available over the network. However, this method increases the complexity of the communication between MPEs as they have to reason about the synchronization part between the servers. Complex algorithms should then be used to be sure that all the MPEs got the necessary information in case of collaborative actions. The complexity and workload distribution rapidly increase when the number of MPEs increases. In case of peer-to-peer networks, MPEs would be able to exchange information directly. This can happen however when the devices are close to each other and can send are receive bigger amount of data. For example, in a smart home, MPEs could be arranged into a peer-to-peer network. The devices could stay connected and be aware of all the other devices in their network.

VI. Case Study: An E-Learning Mobile Application

To better illustrate the behavior of a MPE and the usefulness of having a network of MPEs, we created an example of a possible scenario for its practical implementation. For simplicity, we focused more on describing the possible situations that highlight the benefits of having a network of MPEs and not on the implementation details. All around the world, students are using online e-learning platforms, like Moodle™ [37], for accessing educational content, completing assignments and participating in discussion related to their courses. These e-learning platforms are quite static as they are waiting for instructions or commands from their users. This is why an e-learning application for mobile devices, i.e., smartphones and tables, with an integrated Proactive Engine, would come in hand. We assume that the application would be directly connected with the web platform and would have access to all the data from the student's account on the LMS.

The application would include an advanced mechanism for displaying notifications and questions for the user, provide hints and trigger alarms. Hints would be used for guiding the user, questions for asking for specific instructions, notifications as short messages to inform the user and alarm to alert him/her in case of extraordinary situations or events. On one hand, even though these features have been integrated in other E-Learning applications, the way they are created and handled represents the novelty in this case. On the other hand, they are already addressing some of the major issues when using an online e-learning platform. These issues appear because of the lack of an immediate notification channel between the students or between the students and the professors in case extraordinary situations appear. Certain online platform have an online mechanism for enrolling to an exam, and students often miss these deadlines, resulting in a big problem both for the student and the administration of universities and schools. More issues include missing deadlines for assignments and nonparticipating in forums.

For instance, the scenario of an instructor that has to give an exam on a specific date, at a specific hour and would be late due to traffic is an example of a foreseen event. The logical sensors of the MPE would know that there is an exam approaching soon based on the calendar of the LMS, where the exact date and hour of the exam would be set. The MPE would send alerts to the instructor and he/she could post a short message, via his/her smartphone, on the forum of the course announcing that he/she will be late. Not only will the students be notified of this, but a person from the administration could also alert the students in person if they would not have their device with them. The physical sensors of the MPE would sense that he is moving and so, would adjust the graphical user interface for writing messages.

```
Proactive Rule R001
Description: This Rule is designed to run on each MPE in order to check for new connections in the same network with which the current MPE could share information if they are working on the same assignment.

data acquisition
    conn [] = getConnectionsOnSameNetwork()
activation guards
    conn.size != 0
conditions
    conn.assignment.isStillValid()
actions
    foreach connection in conn []
      if(usersWorkOnSameAssignment(
      connection.assignemnt.ID))
         sendMessageToMPE(conn.ID, message)
         inviteOtherMPEforCollaborativeWork(
         connection.assignemnt.ID)
      end if
    end foreach
rules generation
    if(!activationGuard)
      createRule002(conn.ID, conn.assignment.ID)
    end if
    cloneRule (R001)
```

Figure 5. An example, in pseudo-code, of a Proactive Rule

```
Proactive Rule R002
Description: This Rule is designed to constantly check if the was any update in the deadline or additional documents were added for this assignment.

data acquisition
    connID = getConnectionID()
    assigID = getAssignmentID()
activation guards
    return deadlineIsStillValid(assigID)
conditions
    return contentWasAdded(assigID)
actions
    if( conditions())
         extractAdditionalContent(assigID)
         alertUsersAboutAdditionalContent (users[])
         updateAlarmOfAssignment(assigID)
    end if
rules generation
    if(!activationGuard)
      cloneRule (R002)
    end if
```

Figure 6. A second Proactive Rule in pseudo-code, generated by the first Proactive Rule R001

More advance actions would include setting an alarm for deadlines, putting the events into an integrated calendar, proposing to students to collaborate on solving assignments with other classmates, which are close to their location or , even more, automatically download documents or course material directly to the smartphones of the students. The majority of these actions are not currently provided by any existing LMS and, adding plugins or third party applications will not change the overall behavior of the system. These actions represent collaborative actions between the LMS and the clients. However, MPEs allow another type of collaboration, where each MPE is part of the process. In addition to traditional forms of communication between mobile devices like calls and messages, MPEs are capable of exchanging relevant context information, checking if there are any errors or mistakes on the data acquired from their local sensors, reaching for remote data, sharing resources and of allowing synchronous/ asynchronous communication. This type of innovative collaboration allows MPEs to provide sophisticated real-time services and to support complex mobile applications.

In Fig. 5, an example of a Proactive Rule, which would be used for this case study, is illustrated in pseudo-code. More specifically, this Proactive Rule would run at each iteration of the Rules Engine and would get activated only when there would be at least two MPEs on the same network. Its purpose is to invite the users of the MPEs, in case they are working on the same assignment, to collaborate and share their knowledge. The *proactive* aspect comes from the fact that this situation is anticipated by the MPEs, without any specific intervention or command from the users of the MPEs.

First, Proactive Rules would check for constraints on the LMS to see if the assignment was made as a collaborative assignment or as an individual assignment. In case the assignment is open to cooperation between students, the Awareness Engine would be alerted. It would then be in charge of two main tasks: detecting from the internal virtual sensors of the MPE that a user is actively working on the assignment and discovering other MPEs that are available on the same network and open to collaboration. Then, Proactive Rules, like the one illustrated in Fig. 5, would get activated and would start analyzing and deciding if there were any real-time possibility of collaboration between the two MPEs. If yes, the Adaptation Engine would trigger different actions, like messages to the users to ask them if they want to collaborate remotely or meet and solve the assignment in case the MPE would detect that they are close to each other, e.g., both MPEs would be connected to the same Wi-Fi network in the campus.

In Fig. 6, a second example of a Proactive Rules is given to illustrate that several Proactive Rules can be generated and executed at the same time by a MPE. Rule R002 was activated by the rule R001 for checking, at every iteration of the Rules Engine, if there were any updates in the deadline of assignment or if additional learning material was added on the LMS. In case these events occur, the users are notified immediately via email or via messages sent trough the Notification Manager and, the alarm that was automatically set by the MPE will be updated in order to announce the users in advanced about their deadline. Rule R002 will not activate any other Proactive Rule, it will just clone itself until the deadline of the assignment has been reached.

VII. Other fields of applications for MPEs

The previous case study indicated that MPEs could be used in education, in a computer supported collaborative work scenario. MPEs could also be implemented into application in other domains, which could benefit from the possibility of having multiple networks of systems capable of performing collaborative actions, like medicine, public transportation, tourism, business and many social networks platforms.

In hospitals, for instance, MPEs could be integrated into various medical systems with computing capabilities that would communicate for providing better services to the patients. All these interconnected devices would have access to a variety of sensors, which could lead to a better anticipation of emergencies and a better response from the medical staff. In transportation, for instance, MPEs could help preventing traffic jam, based on the experiences of other MPEs. If the system would detect that multiple devices are moving very slowly on a section where the high speed is allowed it could alert the other MPEs about the situation and an alternative route would be proposed. In tourism, users could experience new ways of connecting with the surrounding environment though the use of MPEs that could obtain relevant information based on their location.

VIII. MPE Implementation

The architecture proposed for mobile devices in this paper is currently under development for the iOS-based devices. A basic working prototype application for smartphones and tablets running an Android Operating System has been already created and is already being tested with basic sets of Proactive Rules. The Rules Engine was implemented using Java and it is capable of running Proactive Rules. The local database and the connection between the Rules Engine and the database were implemented using SQLite™ [38] and ORMLite™ [39]. The MPE is able to communicate with other MPEs via Google's GCM framework [40] that allows devices to exchange messages over the same connection. The Adaptation Engine and the Awareness Engine will run on single background threads that are capable of monitoring the environment, capture data from various sensors that are integrated into mobile devices and performing adaptation at the level of the interface. The Notification Manager is using the notifications built-in libraries of the Android Operating System. The graphical user interface is still subject to modifications, as it depends on the specific requirements to which the application will serve. A complete description of its implementation and of its performance will soon follow, after the prototype will be finalized.

IX. Conclusion and future work

In this paper, we have outlined that we are heading towards various distributed networks of proactive, context-aware and self-adaptive systems. Our reasoning is supported by the continuous intensive research initiatives of many laboratories in the fields of Context-Awareness, Proactive Computing and Self-Adaptation.

Our work opens new substantial research possibilities and new perspectives on how future smart applications will behave, communicate and collaborate. To support our vision, we proposed a model for mobile devices that is able to integrate all the discussed properties. Designers can now focus more on high-level implementation planning and on the functionalities that their applications will provide, than on architectural design, configuration details and on compatibility issues.

Adaptability is provided at a behavioral level, using the model of feedback loops, and at the level of the communication between the different components of our model. Awareness comes from the fact of taking into account different contexts and Proactivity is achieved by using a rules-based engine for handling foreseen events. With the proposed model, the user is focusing more on how to interact with the application and not how to manage and configure the system.

A. Challenges Ahead

Two of the most challenging points in the close future are to develop techniques for communication and collaboration between MPEs and to design and develop smart applications based on the proposed model taking in account important factors like user mobility, different computing capabilities of various devices and privacy issues.

B. Future work

A case-study based evaluation will follow for validating all the characteristics of the presented model and for answering to some research questions such as whether or not the model is correctly providing routines in a context-adaptive manner, or if the parts of the model are really taking into account the user's preferences, or if the model has self-adaptive properties that allows it to modify its behavior. In the upcoming case study, the application presented as the motivating scenario in this paper will be implemented for mobile devices. Proactive Rules will be developed to take care automatically of the tasks that can be performed by the application without asking for specific commands from the user.

Important research is still to be done for exploring the computing capabilities of other intelligent devices that could support proactive context-aware adaptive applications, not only mobile devices. Networks of wearable devices, ubiquitous devices and other computing systems could be thus connected to form a global distributed intelligent network.

References

[1] R. A. Dobrican and D. Zampunieris, "Moving Towards a Distributed Network of Proactive, Self-Adaptive and Context-Aware Systems, " in Proc. of the 6th International Conference on Adaptive and Self-Adaptive Systems and Applications, (ADAPTIVE), May 2014, pp. 22-26.

[2] M. Salehie and L. Tahvildari, "Self-adaptive software: Landscape and research challenges," ACM Transactions on Autonomous and Adaptive Systems, 2009, vol. 4, pp. 1-42.

[3] Y. Brun, G. M. Serugendo, C. Gacek, H. Giese, H. Kienle, M. Litoiu, H. Muller, M. Pezze, and M. Shaw, "Engineering Self-Adaptive Systems through Feedback Loops," in Software Engineering for Self-Adaptive Systems, Lecture Notes In Computer Science, Springer, 2009, vol. 5525, pp. 48-70.

[4] S. Dobson, S. Denazis, A. Fernandez, D. Gaiti, E. Gelenbe, F. Massacci, P. Nixon, F. Saffre, N. Schmidt, and F. Zambonelli, "A survey of autonomic communications," ACM Trans. Auton. Adapt. Syst., 2006, vol. 1, pp. 223-259.

[5] I. Epifani, C. Ghezzi, R. Mirandola, and G. Tamburrelli. "Model evolution by run-time parameter adaptation, " in Proc. of the 31st International Conference on Software Engineering, ICSE 2009, IEEE, pp. 111-121.

[6] IBM Corporation: "An Architectural blueprint for autonomic computing," White paper, 4th edn., IBM Corporation, 2006.

[7] J. Indulska and P. Sutton, "Location management in pervasive systems," in Proceedings of workshop conference on the Australasian information security ACSW frontiers, vol. 21, January 2003, pp. 143-151.

[8] S. Kurkovsky, "Location-dependent and Context-Aware Computing," Next Generation Mobile Networks and Ubiquitous Computing, 2001, 156.

[9] B.H.C. Cheng, R. de Lemos, H. Giese, P. Inverardi,J. Magee, J. Andersson, B. Becker, N. Bencomo, Y. Brun, B. Cukic, G. M. Serugendo, S. Dustdar, A. Finkelstein, C. Gacek, K. Geihs, V. Grassi, G. Karsai, H. M. Kienle, J. Kramer, M. Litoiu, S. Malek, R. Mirandola, H. A. Muller, S. Park, M. Shaw, M. Tichy, M. Tivoli, D. Weyns, and J. Whittle, "Software Engineering for Self-Adaptive Systems: A Research Roadmap," in Software Engineering for Self-Adaptive Systems, Lecture Notes In Computer Science, Springer, 2009, vol. 5525, pp. 1-26.

[10] D. Tennenhouse, "Proactive Computing," Communications of the ACM, 2000, vol. 43, issue 5, pp. 43-50.

[11] A. Oulasvirta and A. Salovaara, "Six modes of proactive resource management: a user-centric typology for proactive behaviors," in Proc. NordiCHI 2004, ACM Press, pp. 57-60.

[12] D. Zampunieris, "Implementation of a Proactive Learning Management System," in Proc. E-learn 2006, AACE Press, pp. 3145-3151.

[13] S. Coronado and D. Zampunieris, "Towards a proactive learning management system using early activity detection," in SITE08, AACE Publishing, 2008, vol. 1, pp. 306-311.

[14] R. Dobrican, S. Reis, and D. Zampunieris, "Empirical Investigations on Community Building and Collaborative Work inside a LMS using Proactive Computing," in Proc. E-learn 2013, vol. 1, pp. 1840-1852.

[15] R. Dobrican and D. Zampunieris, "Supporting collaborative learning inside communities of practive through proactive computing," in Proc. EDULEARN13, 2013, pp. 5824-5833.

[16] D. Shirnin, S. Reis, and D. Zampunieris, "Experimentation of Proactive Computing in Context Aware Systems: Case Study of Human-Computer Interactions in e-Learning Environment," IEEE CogSIMA, Feb. 2013, pp. 269-276.

[17] V. C. Ostuni, T. Di Noia, R. Mirizzi, D. Romito, and E. Di Sciascio, "Cinemappy: a Context-aware Mobile App for Movie Recommendations boosted by DBpedia," in SeRSy, October 2012, pp. 37-48.

[18] P. Coppola, V. Della Mea, L. Di Gaspero, S. Mizzaro, I. Scagnetto, A. Selva, L. Vassena, and P. Riziò, "MoBe: context-aware mobile applications on mobile devices for mobile users," in Proc. of the International Workshop on Exploiting Context Histories in Smart Environments, 2005, Munich, Germany, pp. 8-13.

[19] E. Bertou and S. Suleman, "OnRoute: A Mobile Context-Aware Public Transportation Planning Application," in HCI International 2013-Posters' Extended Abstracts, Springer Berlin Heidelberg, 2013, pp. 299-303.

[20] J. Yim and C. Le. Thanh, "Museum Guide, a Mobile App," in Computer Applications for Software Engineering, Disaster Recovery, and Business Continuity, Springer Berlin Heidelberg, 2012, pp. 36-41.

[21] D.B. Abeywickrama, N. Bicocchi, and F. Zambonelli, "SOTA: Towards a General Model for Self-Adaptive Systems," 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises (WETICE), IEEE, June 2012, pp. 48-53.

[22] J. Hakkila, and J. Mantyjarvi, "Collaboration in context-aware mobile phone applications," in Proc. of the 38th Annual Hawaii International Conference on System Sciences, 2005. HICSS'05, IEEE, 2005, pp. 33a-33a.

[23] B.H.C. Cheng, R. de Lemos, P. Inverardi, and J. Magee (Eds), "Software engineering for self-adaptive systems," in Dagstuhl Seminar, 2009, vol. 5525.

[24] W. Heaven, D. Sykes, J. Magee, and J. Kramer, "A case study in goal-driven architectural adaptation," in Software Engineering for Self-Adaptive Systems, 2009, Springer Berlin Heidelberg, pp. 109-127.

[25] A. A. Mansor, W. M. W. Kadir, and H. Elias, "Policy-based approach for dynamic architectural adaptation: A case study on location-based system," in 5th Malaysian Conference on Software Engineering (MySEC), IEEE, December 2011, pp. 171-176.

[26] M. M. Islam, M. A. Sattar, M. F. Amin, X. Yao, and K. Murase, "A new constructive algorithm for architectural and functional adaptation of artificial neural networks," in Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on, vol. 39, no. 6, 2009, pp. 1590-1605.

[27] Z. Yang and Z. Jin, "Modeling and Specifying Parametric Adaptation Mechanism for Self-Adaptive Systems," in Requirements Engineering, Springer Berlin Heidelberg, 2014, pp. 105-119.

[28] G. Tamura, N. M. Villegas, H. A. Müller, L. Duchien, and L. Seinturier, "Improving context-awareness in self-adaptation using the DYNAMICO reference model," in Proc. of the 8th International Symposium on Software Engineering for Adaptive and Self-Managing Systems, IEEE Press, 2013, pp. 153-162.

[29] H. Klus, D. Niebuhr, and A. Rausch, "A component model for dynamic adaptive systems," in International workshop on Engineering of software services for pervasive environments: in conjunction with the 6th ESEC/FSE joint meeting, ACM, 2007, pp. 21-28.

[30] R. Mizouni, M. A. Serhani, A. Benharref, and O. Al-Abassi, "Towards Battery-Aware Self-Adaptive Mobile Applications," in Proc. of the 9th International Conference on Services Computing (SCC), IEEE , June 2012, pp. 439-445.

[31] A. Karadimce and D. C. Bogatinoska, "Using hybrid mobile applications for adaptive multimedia content delivery," in Proc. of the 37th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO), May 2014, pp. 686-691.

[32] T. Ruiz-López, C. Rodríguez-Domínguez, M. J. Rodríguez, S. F. Ochoa, and J. L. Garrido, "Context-Aware Self-adaptations: From Requirements Specification to Code Generation," in Ubiquitous Computing and Ambient Intelligence. Context-Awareness and Context-Driven Interaction, Springer International Publishing, 2013, pp. 46-53.

[33] M. Hussein, J. Han, and A. Colman, "An Approach to Model-Based Development of Context-Aware Adaptive Systems," in Proc. of the 35th Annual Computer Software and Applications Conference (COMPSAC), July 2011, IEEE, pp. 205-214.

[34] L. David, M. Endler, S.D.J. Barbosa, and J.V. Filho, "Middleware Support for Context-Aware Mobile Applications with Adaptive Multimodal User Interfaces," in Proc. of the 4th International Conference on Ubi-Media Computing (U-Media), July 2011, pp. 106-111.

[35] M.T. Segarra and F. Andre, "Building a Context-Aware Ambient Assisted Living Application Using a Self-Adaptive Distributed Model," in Proc. of the 5th International Conference on Autonomic and Autonomous Systems, ICAS '09, April 2009, pp. 40-44.

[36] J. Yau and M. Joy, "A Context-aware and Adaptive Learning Schedule framework for supporting learners' daily routines," in Proc. of the 2nd International Conference on Systems, ICONS '07, April 2007, pp. 31-37.

[37] Moodle - Modular Object-Oriented Dynamic Learning Environment. [retrieved: April, 2014]. Available from: https://moodle.org/

[38] SQLite Framework. [retrieved: April, 2014]. Available from: http://www.sqlite.org/

[39] ORMLite - Lightweight Object Relational Mapping (ORM) Java Package. [retrieved: April, 2014]. Available from: http:// http://ormlite.com/

[40] Google GCM – Google Cloud Messaging for Android. [retrieved: August, 2014]. Available at http://developer.android.com/google/gcm/index.html