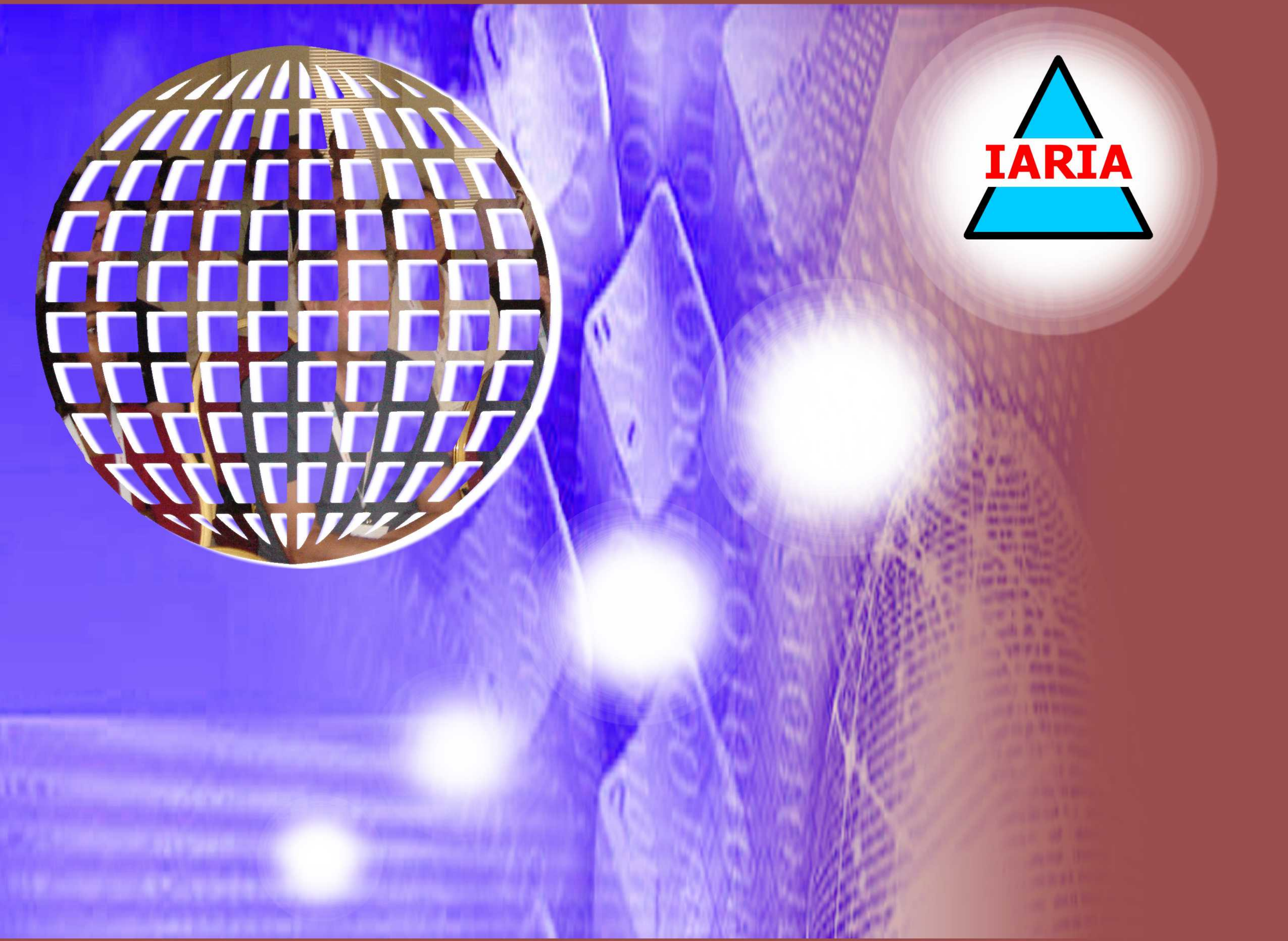
International Journal on Advances in Networks and Services



2020 vol. 13 nr. 1&2

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 13, Number 1 & 2, 2020

International Journal on Advances in Networks and Services
Volume 13, Numbers 1 & 2, 2020

CONTENTS

Using Rule-Based Decision Trees for Automatic Passive Diagnostics of the Network Problems

Martin Holkovič

Flowmon Networks
Sochorova 3232/34
Brno 61600, CZ
Email: martin.holkovic@flowmon.com

Ondřej Ryšavý

Faculty of Information Technology
Brno University of Technology
Brno 61266, CZ
Email: rysavy@fit.vutbr.cz

***Abstract*—Network troubleshooting often requires a detailed analysis that may involve network packet capturing and a manual analysis using tools such as Wireshark. This is time-consuming and requires deep knowledge of communication protocols. Therefore, this domain is a suitable candidate for the deployment of an expert system. In this paper, we consider a rule-based system integrating the expert knowledge that performs an automatic root cause analysis of network problems identifiable from network communications. The system is open, thus it is possible to add new rules as needed, e.g., for specific and recurring cases of a target environment. The rules are evaluated in a tree-based fashion, which enables us to collect additional information during the problem search to better explain the possible causes. We successfully deployed the tool as part of a commercial tool for network monitoring.**

Keywords*–*Network diagnostics; rule-based diagnostics; fault tree analysis; event-based diagnostics; decision trees.

I. Introduction

Network infrastructure and applications are complex, prone to cyber attacks, outages, performance problems, misconfigurations, and problems caused by software or hardware incompatibility. All these problems may affect network performance and user experience [2], which may have fatal consequences for critical network infrastructure, e.g., e-health, e-government, Industrial IoT, smart grid, etc. Network troubleshooting is thus among the most common and important activities by network administrators. Despite the help of the current network monitoring tools, identification of a root cause of issues can be a complicated and mostly manual activity. The tools often reveal symptoms of the problem but the reasoning and problem localization are left for human operators expecting that they understand the problem and have sufficient knowledge of the technologies involved. Even if it is the case, the troubleshooting can be a lengthy and tiresome process that requires inspection of different sources of information, e.g., log files, the content of various tables, communication traces, etc. Application communication protocols are designed to implement the data exchange of remote parties. The protocol specification defines the syntax and meaning of messages, the way the conversation is controlled, and also the indication of error states. Thus, by inspecting the network communication it is possible to understand the situation and identify the indicated errors and in many cases also their probable cause.

Unfortunately, many network administrators do not have the proper tools and/or knowledge to diagnose and fix network problems effectively, and they require an automated tool to diagnose these errors [3]. Zeng et al. [4] provide a short survey on network troubleshooting from the administrators' viewpoint identifying the most common network problems: *reachability problems, degraded throughput, high latency, and intermittent connectivity*. The consulted network administrators expressed the need for a network monitoring tool that would be able to identify such problems.

This paper proposes a system, which creates diagnostic information only by performing passive network traffic packet-level analysis. Previous research and development provided tools for helping administrators to diagnose faults [5] and performance problems [6]. However, these tools either require *installation of agents on hosts, active monitoring, or providing rich information about the environment*. The idea behind our proposal is to automate the reasoning usually done by network analysts when investigating the root cause of an error from the captured network traces. It means that it is not necessary to change the network environment nor deploy any new devices. The troubleshooting process may remain unchanged except that one of the most labor-intensive parts represented by the packet-level traffic analysis is automated. Still, the user can verify the results obtained from the automated analysis as the process provides sufficient diagnostic information for the identification of problem relevant artifacts.

One of the most common ways of analyzing network traffic is by using a network packet analyzer (e.g., Wireshark). The analyzer works with captured network traffic (PCAP files) and displays structured information of layered protocols contained in every packet (encapsulated protocols, protocol fields). Administrators work with this information, check transferred content and compare the data with expected values. This process, done manually, is time-consuming and requires a good knowledge of network protocols and technologies.

The main contribution of this paper is a proposal of a tool for automatic diagnoses of network related problems from network communication only. Our approach tries to imitate a diagnostic process of a real administrator using the fault tree method and a popular packet parsing tool TShark. We have also implemented a proof-of-concept implementation to confirm the viability of the approach. This paper is an extension of our previous paper [1]. The most significant change is the improvement of input data processing. A new more efficient mechanism of converting input data into a specific indexable format has been implemented. This change required significant modification of the method the system uses to access the data. However, the new format simplifies processing of other data types and reduces the execution time of the whole diagnostic significantly. A simple example of a tool usage for another data type (log files) is also presented.

The paper is organized as follows. Section II defines the problem statement and research questions. Section III discusses related work and describes diagnostic approaches. Our solution consists of five stages and is introduced in Section IV. Section V instructs network administrators how to use our system and shows how we model diagnostic knowledge. Section VI shows the output from the tool and evaluates the performance. Finally, Section VII is the conclusion, which summarizes the current state and proposes future work.

II. Research Questions

Our primary goal is to design a system that infers possible causes accountable for network related problems, such as service unreachability or application errors. Offering a list of actions for fixing the errors' cause is the secondary and optional goal. All this information is gathered only from captured network communication, which makes this approach applicable to various scenarios.

In our work, we focus on enterprise networks that have complex networking topologies, usually consisting of various network and end-point devices. The availability of network traces in the form of packet captures is essential to our method. Thus, we expect that administrators can collect network communication at appropriate locations in the network. Also, we consider that the capturing process creates packet captures without packet losses. As this may be difficult to guarantee for high-speed networks without using specialized hardware, for the diagnostic we usually do not require all communication. Thus, the packet capture can be recorded by applying a suitable filter to reduce the amount of data that needs to be processed.

To achieve our goal, we need to find answers to the following research questions:

1) How to model different network faults in a suitable way for implementation in a diagnostic system? *Reachability, application specific, and device malfunctioning problems* can cause various networking issues. We need to have a unified approach for modeling these problems to identify the symptoms and link them with root causes.
2) What information should be extracted from the captured network communication to identify symptoms of failures? In our case, we can passively access the communication in the monitored network and extract the necessary data to detect possible symptoms. An approach that can efficiently detect the symptoms in terms of precision and performance is needed.
3) How to identify the root cause of the problem, if we have a set of related symptoms? The core part of the diagnostic engine is to apply knowledge gathered from observed symptoms to infer the possible root cause of the problem. The result should provide the information in sufficient detail. For instance, if the authentication during the establishing of the connection fails, then we would like to know this specific information instead of a more general explanation (e.g., unable to establish a connection).
4) What actions can be provided to the administrator to fix the problems? Based on the observed symptoms and the root cause, the system should be able to provide fixing guidelines. These guidelines are supposed to be easy to understand even for an inexperienced administrator.

III. Related Work

A lot of research activities were dedicated to the diagnoses of network faults. Various methods were proposed for different network environments [5], in particular, home networks [7], enterprise networks [8]–[11], data centers [6], backbone and telecommunications networks [12], mobile networks [13], Internet of Things [14], Internet routing [15] and host reachability. Methods of network troubleshooting can be roughly divided into the following classes:

Active methods use traffic generators to send probe packets that can detect the availability of services or check the status of applications [16]. Usually, generators create diagnostic communication according to the test plan [8]. The responses are evaluated and provide diagnostic information that may help to reveal device misconfiguration or transient fail network states. Diagnostic probes introduce extra traffic, which may pose a problem for large installations [11]. Also, active methods may rely on the deployment of an agent within the environment to get information about the individual nodes [9].

Passive methods detect symptoms from existing data sources, e.g., traffic metadata [12], traffic capture files, network log files [15], performance counters. Passive methods can utilize the data commonly provided by various network monitoring systems.

Some systems combine passive traffic monitoring to detect faults with active probing to determine the cause of failure. Identifying anomalies related to network faults and linking them with possible causes commonly utilizes some of the following approaches:

Inference-based approach uses a *model* to identify the dependence among components and to infer the faults using a collection of facts about the individual components [9], [17].

Rule-based approach uses *predefined rules* to diagnose faults [10]. The rules identify symptoms and determine how these contribute to the cause. The rules may be organized in a collaborative environment for sharing knowledge between administrators [7]. Kim et al. [18] propose a rule-based reasoning (RBR) expert system for network fault and security diagnosis. The system uses a set of agents that provide facts to the diagnostics engine. De Paola et al. [19] deals with a distributed multi-agent architecture for network management. The implemented logical inference system enables automated isolation, diagnosis, and repairing network anomalies through the use of agents running on network devices. Dong and Dulay [20] developed an assumption-based argumentation to create an open framework of the diagnosis procedures able to identify the typical errors in home networks. Rule-based systems often do not directly learn from experience. They are also unable to deal with new previously unseen situations, and it is hard to maintain the represented knowledge consistently [5].

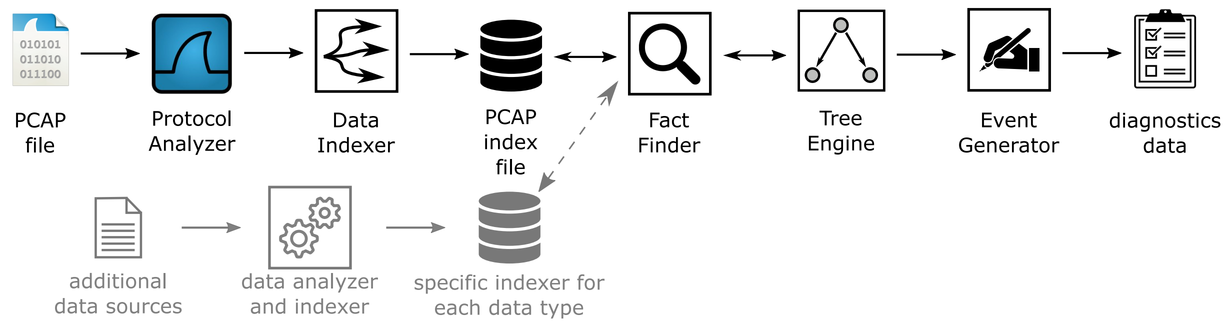


Figure 1. The top-level architecture of the proposed system. The architecture consists of five stages and one intermediate data storage (index file). The grey area represents optional architecture extensions — additional data sources.

Classifier-based approach *requires training data* to learn the normal and faulty states. The classifier can identify a fault and its likely cause [21]. Classifier-based methods were considered for misconfiguration detection in the home networks [22] and in the large network infrastructure [23]. *Tranalyzer* [24] is a flow-based analyzer that does traffic mining and a statistical analysis for large-scale networks. Big-DAMA [25] is a novel framework for detection and diagnosis of network traffic anomalies.

Network diagnostics based on traffic analysis can also use methods proposed for anomaly detection as some types of faults result in network communication anomalies.

Compared to other rule-based solutions, our system uses decision trees, which allows us to define more complex situations. Compared to simple rules (as used, for example, by a fishbone diagram), it is possible to make decisions based on previous diagnostic steps. Another difference is that our system does not need to know in advance what is wrong or what to focus on. Also, our system is not limited to only one type of data, and diagnostic rules are understandable by real administrators (not just scientists and programmers).

IV. Proposed System Architecture

We have built an expert system for analyzing network traffic, that has already been integrated into a worldwide business product [26]. The system combines rule-based and inference-based methods as it is easy to understand for network administrators. While the use of classifier-based methods has been proven very suitable for anomaly detection it lacks the capability to provide additional information for the detected case. The advantage of learning from provided data can only be exploited if a large set of annotated data is available. Contrary, the rule-based method can be extended also for detecting rare cases. The system only requires captured network traffic containing enough information about the event. Thus it is a completely passive method. Active methods generate additional traffic into a network (which can be unwanted in some situations) and require access to the network.

The proposed system is a processing pipeline that consists of several stages, as shown in Figure 1. The first stage, labeled as *Protocol Analyzer*, filters and decodes input packets using an external tool. The second stage takes decoded packets and converts them into a format for easier and faster data access (PCAP index file). The third stage, named *Fact Finder*, executes simple rules to identify facts significant from the diagnostics point of view. In the fourth *Tree Engine* stage, the decision tree utilizes the *Fact Finder* and identifies the possible problem cause. The fifth, and the last, stage *Event Generator* generates diagnostic outputs that contain detected errors and suggested solutions. Stages three, four, and five are easily extendable by the administrator who can add new rules and definitions.

The system can also be extended to use different data sources (e.g., log files or NetFlow records), as shown on the second row in Figure 1. Each data source requires specific data preprocessing that leads to the creation of an index file. The common part starts with the *Fact Finder* that can search indexed data of different data sources. If not specified otherwise, in the rest of the paper, we describe and evaluate the system only for a single data source represented by captured packet traces.

A. Protocol Analyzer

The first step in the processing pipeline is decoding captured network traffic in the PCAP format into a readable JSON format. We employ the tool TShark, which is a command-line version of the widely-used network protocol analyzer Wireshark. Because TShark follows the field naming convention used by Wireshark, we can use Wireshark Display Filter Expressions to select packet attributes. TShark supports all packet dissectors available in Wireshark. An example of TShark's output format with some omitted data is displayed in Figure 2.

```
{
...
  "_source": {
    "layers": {
      "frame": {
        "frame.number": "15",
...     "frame.len": "84",
      "ip": {
        "ip.ttl": "50",
        "ip.proto": "6",
...
      "tcp": {
        "tcp.srcport": "25",
...     "tcp.dstport": "1470",
      "smtp": {
        "smtp.response.code": "235",
...
}
```

Figure 2. An output from the TShark's JSON format.

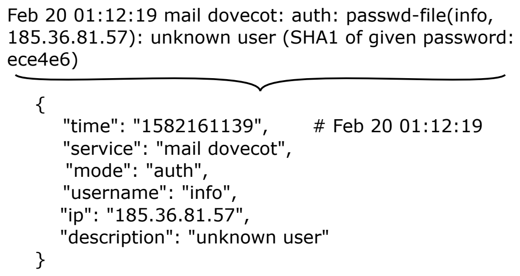
Using TShark brings the following benefits:

- many protocol dissectors are available and the community quickly provides a parser for an emerged protocol;
- tunneled, segmented and reassembled data are support;
- data presentation is consistent with the Wireshark, which allows the creation of an easy-to-read API for diagnostics.

TShark provides not only data of fields from supported network protocols but also some computed data, such as round trip time, missing or retransmitted TCP segments, which can be used in diagnostic rules.

Even if our primary use case is to diagnose problems inside the captured network data, we would like to test that our system can work with other data sources as well. For this test, we have chosen to use the log files. Because each application has its own format of log messages and we were not able to find a universal tool that can parse the content of any log message into a JSON object, we have implemented a custom parser.

Our data preparation script takes log records one by one, and if a record matches some of the predefined regular expressions, the record is converted into a JSON format, as shown in Figure 3. Currently, only a few applications are supported - postfix, dovecot, and fail2ban. The output JSON format has the same structure as the JSON from the TShark tool, so future processing will remain the same.



```
Feb 20 01:12:19 mail dovecot: auth: passwd-file(info,
185.36.81.57): unknown user (SHA1 of given password:
ece4e6)
```

```
{
   "time": "1582161139",      # Feb 20 01:12:19
   "service": "mail dovecot",
   "mode": "auth",
   "username": "info",
   "ip": "185.36.81.57",
   "description": "unknown user"
}
```

Figure 3. Conversion of a single log record into a JSON object.

B. Data Indexer

Data Indexer converts data from the JSON format into a format suitable for fast searching by packets' field names (attributes) and their values. Most of the time, it will not be necessary to process the packets one by one, which significantly improves the resulting diagnostic speed. Each input packet is indexed and the following data is stored:

1) the packet itself;
2) a set of all field names of the packet;
3) map of values assigned to each packet's field.

Figure 4 shows an example of how indexing works. The entire index is represented by an associative array. First, the packet is stored under the *_raw* key and a packet number (in this case, 3). Using this value, it is possible to retrieve the packet in the same format as returned by TShark. Subsequently, the packet number is stored under a set of all indexed packets stored under the *_packets* key. This set will simplify some operations, and its usage can be seen in Figure 5. In the next steps, for each packet attribute and its values (each attribute can contain multiple values) a set of packet numbers is created (when it does not already exist). After that, the current packet number is added to the set.

```
index = dict() # associative array in
    Python
index["_raw/3"] = {"frame.number":
    ["3"], "dns.id": ["0x00007df5"],
    "ip.addr": ["192.168.1.1",
    "192.168.1.100"],...} # under the
    _raw + packet number key, the
    original packet in the JSON format
    is stored
index["_packets"] = {1, 2, 3} #
    _packets index contains set of all
    packet numbers
index["dns.id"] = {3} # all packets
    with any "dns.id" field value are
    saved under the field name key
index["dns.id/0x00007df5"] = {3} #
    packets which contain "dns.id" field
     with value "0x00007df5" are saved
    under field name/field value key
index["ip.addr"] = {1, 2, 3}
index["ip.addr/192.168.1.1"] = {1, 3}
index["ip.addr/192.168.1.100"] = {3}
```

Figure 4. An example of the indexing of a few fields from the DNS packet. The bold text is showing index keys and values, which have been added because of the new packet.

C. Fact Finder

The *Fact Finder* aims to identify specific situations useful for network diagnostics. Facts can be attributed to one or more packets, which are in some relation. For example, a successful DNS name resolution is a fact that consists of a query and a corresponding reply DNS messages. The facts are specified by rules describing which packets should be found and which relation they should fulfill. The format of these rules is described in Subsection V-B. A rule can consist of up to three parts:

1) a list of packet filters;
2) a list of assertions to express relation constraints;
3) parameters for the filters and assertions.

The system evaluates rules as follows: (i) Parameters are replaced by provided values. (ii) Each packet filter returns a list of packets matching the filter. (iii) Assertions are evaluated to select sets of packets satisfying the constraints. A result has the form of a collection of sets of packet numbers, e.g., a rule that identifies DNS request-reply pairs checks that the transaction ID in both the request and reply packets match. The last step is converting sets of packet numbers into lists of packets. Packets in lists are ordered by the packet numbers.

Filter expressions use Wireshark's display filter language. By using this language, the expression can be first tested in Wireshark before it is used in a *Fact Finder* rule. Assertion constraints use our created language that is based on the Wireshark's display filter language. There are three changes made to the original language, which add support of:

1) working with packets from filter expressions;
2) simple math operations (+, -, *, /);
3) parameters for expressions. The parameters do not increase language capability but aim to simplify rule definition.

The evaluation of the facts begins with searching for packets with specific attributes. However, this varies depending on how these attributes are specified. In the case of a simple condition, it is possible to use the index created in the *Data Indexer* step, but with more complex conditions, this is not possible. A more complex condition is one that contains either a regular expression, function (string length, substring), or some comparison ($<, <=, >, >=$).

If it is possible to search for packets using the created index, the appropriate packet numbers are searched for using each attribute specified. Based on the specific relation between attributes, the adequate set operation is applied to the sets of packets. This process is shown in Figure 5. This figure describes finding packets by using the created index.

```
search: dns
result = index["dns"]

search: dns.flags.response == 1
result = index["dns.flags.response/1"]

search: dns and ip.addr == "10.10.10.1"
dns_packets = index["dns"]
ip_packets=index["ip.addr/10.10.10.1"]
result = dns_packets.intersection(
    ip_packets) # packets both in
    dns_packets and ip_packets

search:smtp.response.code != 250
all_packets = index["_packets"]
skip_packets = index["smtp.response.
    code/250"]
result = all_packets.difference(
    skip_packets) # packets in
    all_packets but not in skip_packets
```

Figure 5. An example of index usage when searching for packets that meet the specified constraints. When combining attributes in constraints, results from individual attributes are combined using set operations. The bold text shows the packet specification.

In case the packet specification cannot be evaluated using the created index, it is necessary to go through each packet and evaluate the condition for its values. This is accomplished by replacing the attributes in the expression (e.g., *smtp.response.code matches "[45] [0-9] [0-9]" and ip.addr = "10.10.1.1"*) with values from each packet. Because a list of values represents each attribute's value, the evaluation process must try all value combinations. If at least one combination fulfills the packet specification, the packet is added to the set of fulfilling packets. The principle is shown in Figure 6. Because there was not such a complicated rule in DNS protocol, we are showing this principle on the SMTP rule.

After all packets have been searched, they are represented only by a list of packet numbers. Before working with the packet's data, it is necessary to replace these packet numbers with the actual packets. After that, constraints defining packet relationships can be evaluated (assert rules). An example of a relationship is a request-reply pair of packets that are linked together by a request ID. Searching for such packets is accomplished by creating all possible packet combinations (Cartesian product) and evaluating all conditions for each combination.

```
search packets: smtp.response.code
    matches "[45] [0-9] [0-9]" and
    ip.addr = "10.10.1.1"
import re # regular expression module
result = set()
def check_packet(packet):
  values = {}
  for value in packet["smtp.response.
      code"]: # only packets with field
      smtp.response.code are used
    values["smtp.response.code"]=value
    for value in packet["ip.addr"]: #
        only packets with attribute ip.
        addr are used
      values["ip.addr"] = value
      if re.search(values["smtp.
          response.code"],"[45][0-9]
          [0-9]") and values["ip.addr"]
          == "10.10.1.1":
        result.add(packet_number)
        return result

for packet_number in index["_packets"]:
  packet = index["_raw/"+packet_number]
  result = check_packet(packet, result)
return result
```

Figure 6. An example of finding all packets that meet the specified condition, which can not be evaluated by using the created indexes. When evaluating a condition, all value combinations are tested for each packet. The bold text shows the packet specification and the corresponding condition.

The principle of evaluating the assert conditions is shown in Figure 7. The code in the figure contains a $relation()$ function that combines all possible values (similar to the code in Figure 6) and compares whether at least one value combination meets the defined relation function (e.g., " == " for equality). The $relation()$ function works with a $packets$ dictionary that contains a list of packets that are saved under the keys defined in the facts section of the rule.

```
facts:
  dns_query: dns.flags.response == 0
  dns_reply: dns.flags.response == 1
asserts:
 - dns_query[udp.stream] ==
    dns_reply[udp.stream]
 - dns_query[dns.id] ==
    dns_reply[dns.id]

result = []
for query in packets["dns_query"]:
  for reply in packets["dns_reply"]:
    if relation(query["udp.stream"],
        "==", reply["udp.stream"])
and relation(query["dns.id"],"==",
    reply["dns.id"]): # relation()
    function checks all combinations of
    values from two lists
      result.append({"dns_query": query
        , "dns_reply": reply})
return result
```

Figure 7. Example of a packet set search (DNS query and response) that meets the defined constraint (packets from the same UDP stream and the same request ID). The bold text is sharing assert constraints and the corresponding $relation()$ function.

D. Tree Engine

The tree engine infers the possible error cause by evaluating a decision tree that contains expert knowledge about supported network protocols and services. Each node of the tree contains a diagnostic question. Questions refer to facts identified by the *Fact Finder*. Based on the question's result, the next tree node is chosen. This node transition creates a path that begins in a root node and finishes in a leaf node. Paths in the tree represent gathered knowledge and lead to the possible cause of the problem.

The decision tree consists of declarative specifications of tree nodes enriched by Python code. The declarative part is responsible for creating the tree and consists of a rule name, a rule type, a *Fact Finder* rule, and two branches, which cover the success and the fail result of the *Fact Finder*. Both branches can define the next rule, which should be processed.

Python codes are located inside the success and fail branches. These codes are responsible for processing logic (e.g., saving packets for future tree nodes or translating error codes from packets into human-readable format) and generating diagnostic results. The format of rules is described in Subsection V-A.

E. Event Generator

During the diagnostic process, a report is created to provide diagnostic information for network administrators. The diagnostic report is produced in a human-readable format, as well as in a machine format useful for further processing or visualization. The report consists of events that are constructed in tree nodes based on the derived knowledge and processed packets. Each event describes one situation that happened in the network. For example, the connection to the HTTP server has been detected.

Each rule consists of a name, description, and severity of the detected situation. Additionally, the event may include a suggestion message and data from the provided packet. The provided packet is specified as a parameter in the tree rule. By using this packet, parameters such as flow identification or timestamp can be associated with the event. Subsection V-C describes the format of the event rules.

V. Rule Specification

The diagnostic engine defines each protocol as a decision tree. The tree consists of nodes representing administrator questions, and edges representing answers to these questions. The edge can move the diagnostic process from one question to another (within the same protocol or another) or finish the process with the discovered result.

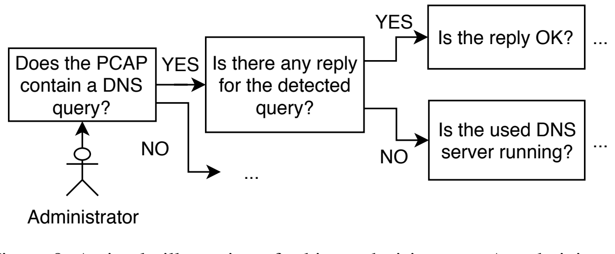

Does the PCAP contain a DNS query?
YES
Is there any reply for the detected query?
YES
Is the reply OK?
...
NO
Is the used DNS server running?
...
NO
...
Administrator

Figure 8. A simple illustration of a binary decision tree. An administrator diagnoses a DNS problem by anwering questions in the predefined order.

The questions simulate thinking of a real administrator. Typically, an administrator starts to search for certain network packet values and after the search for them is finished, the administrator searches for next values based on the result. In our solution, each question can only have two answers: success or failure. This yields a binary decision tree. Figure 8 shows an example of a small portion of the DNS tree.

The decision tree needs to be converted to a format understandable by our system. This conversion is split into three steps: 1) defining tree nodes (*Tree node* rules), 2) defining conditions for choosing tree nodes (*Fact Finder* rules) and 3) defining the diagnostic report (*Event definition* rules). The following subsections describe the syntax for each of these rules. The reason why a node rule does not contain a lookup code and an event definition directly and they need to be defined in separate rules is that multiple rules would not be able to use the same lookup code and events (increases reusability).

The conversion of the decision tree assigns a name to each tree node. We use the node names as labels for switching from one node to another. Each node tries to find specific facts, defined as a *Fact Finder rule*. Based on the condition, if some fact was found or not, the next diagnostic step is chosen. Each rule can have one or none success and fail branches. Branches contain executable Python code and the next node rule name. After the execution of the Python code, the analysis switches to the next node. Figure 9 shows the pseudocode for writing tree nodes.

```
tree_node_id:
  if fact_finder_rule finds some facts:
    success branch_code
    jump to the next tree_node_id
  else:
    fail branch_code
    jump to the next tree_node_id
```

Figure 9. Pseudocode for writing a tree node. Each node should have a unique id, lookup condition, and branch codes.

A. Tree Node Rules

All the rules are saved in a declarative YAML format. This format is easily understandable by programming code and by people without programming skills (we assume that not all network administrators are also programmers). Even if the system already contains some protocols, the administrators can easily add new protocols or can extend capabilities of the current protocols by updating the rules. In the following paragraphs, the format of the rules will be described. Names of the sections as they used in rules are placed inside the text.

The rule definition begins with the rule name (rule section *id*) and the execution of a *Fact Finder* rule (rule section *query*). The result of the *Fact Finder* is a list of associative arrays. Each array can contain multiple packets, where the packet name is the key to the array. These packets will be processed according to the *Tree Node* rule type (rule section *type*). The default behavior selects the first array from the list of arrays (the list is ordered by the arrival time of packets) and marks it as a found fact. The second type of rule is a "foreach" type, which iterates through the list of arrays and progressively marks each array as a fact and executes the defined rule. For example, the foreach rule can analyze each query to the server or each response to the selected query (as shown in Figure 10).

```
id: DNS query detected # name of the
    rule
query: exists DNS reply for the detected
     query? # Facts Finder rule
type: foreach
success:
 state: is reply ok? # next state
 code: | # Python code follows
  reply_pkt = _fact["dns_reply"]
  save("dns_reply", reply_pkt)
  event("reply_detected", reply_pkt)
fail:
 state: find any reply from the same
     destination server # next state
 code: | # Python code follows
  query_pkt = load("dns_query")
  event("reply_not_detected",query_pkt)
```

Figure 10. Simple Tree Engine rule showing what should be done if a DNS query was detected.

Furthermore, the rule consists of two parts, with only one executed (depending on whether the diagnostic engine has found the searched fact or not). The format of both parts is the same. Each part consists of the name of the next rule with which the diagnostics should continue (rule section *state*) and the Python code (rule section *code*). Each rule can switch to to a rule from another protocol to diagnose problems across several protocols, e.g., if an SMTP communication is not detected, we will check if there are any ICMP unreachable messages, failed TCP connection attempts or incorrect DNS resolutions. If the next rule is not specified, the diagnostic engine stops the diagnostic process.

The Python code can process packet data, make logical decisions and most importantly, generate diagnostic output. Within the Python code, it is possible to use any Python 3 code and it is also possible to utilize the following variables and functions defined by the engine:

1) *fact* - contains the first fact found (or the next one in the foreach type rule)
2) *facts* - contains all the facts found
3) *save()* - saving any value for further processing (inside another Tree node rule or as a parameter in the *Fact Finder* rule);
4) *load()* - read the value previously saved by the save() function;
5) *event()* - generates a diagnostic report, where the parameter is a packet to which the report refers.

Figure 10 shows an example of a rule defining the middle node from the tree in Figure 8. The figure shows a node describing that a DNS query has been detected (*id*) and the rule is looking for a DNS response for the detected query (*query*). For each detected reply, a successful section (*success*) is executed (foreach *type*). The response is saved, the diagnostic message is generated (*code*), and the diagnostic process continues to check whether the response is without error (*state*). If no response to the query is found, the failure section is executed (*fail*). First, the original query is retrieved, the diagnostic message is generated (*code*), and then the diagnostic process continues with the next state (*state*).

B. Fact Finder Rules

Rules in this section describe how a question is converted into packet lookup functions. Each rule may look for several independent packets, which are combined and checked if their relation fulfills assert conditions. Each question returns a list of associative arrays, where each arrays represents a unique combination of packets fulfilling the assert conditions (packet names are arrays keys).

Each rule needs to have a name (rule section *id*), which is used inside the *Tree Node* rules. The rule can have parameters to make rules more reusable (rule section *params*). For example, instead of creating a rule for each error code, it is possible to create one rule with a parameter containing the expected error code. Parameters are used as variables in the following sections and can have a format of a single value or single packet (previously saved by *save()* function within the *Tree Node* rule). Rule section *facts* define the name of the searched packets and the filter. The filter uses Wireshark display language and specifies which packets should be assigned to the specified name (each packet can be assigned to multiple packet names). Rule section *asserts* define conditions for the detection of packets. The conditions use our custom language, which is based on the Wireshark display language. However, the custom language allows to:

- use math operations addition(+), subtraction(-), multiplication(*) and division(/);
- use single value parameters as if the values were directly inserted into the condition;
- use values from packets (provided from packets or params section). The format is packet_name[field.name], where field.name is the Wireshark name assigned to the attribute.

Figure 11 shows an example of a simple rule for the question *Is there any DNS reply for the detected DNS query?* After the rule name (*id*), the parameter *dns_query* for the assert conditions is specified (*params*). The rule contains a definition of the *dns_reply* packet (*packets*), which is used in the assert conditions. The conditions (*asserts*) are checking whether the *dns_reply* belongs to the same UDP stream as the provided *dns_query* and if the reply packet is answering to the specific query.

```
id: exists DNS reply for the detected
   query?  # name of the rule
params: # saved data from any tree rule
 −dns_query
facts: # which packets we are looking
   for
 −dns_reply: dns.flags.response == 1
asserts: # packets relation constrain
 −dns_query[udp.stream]==dns_reply[udp.
    stream]
 −dns_query[dns.id]==dns_reply[dns.id]
```

Figure 11. Example of a DNS fact rule for checking if the PCAP file contains a reply for the provided query or not.

By default, the *Fact Finder* rules are working with the data saved inside the PCAP index file. To allow searching for facts from different data index files, it is necessary to specify the data type (rule section *type*). Figure 12 shows an example of such a rule that looks for a log record containing specific values related to the wrong username event. Because the rules are using the same *Tree Node* engine, the rule of one type can use parameters from a different type of rule. In our example, we are looking for a record with the same IP address as used in the provided *imap_query* packet.

```
id: wrong username?
params: # saved packet from previous
    rules
 -imap_query
type: log # the data are saved in
    different index file
facts:
 -auth: description == "unknown user"
     and ip == imap_query[src.ip]
```

Figure 12. Fact rule for different data source (log file). The rule is checking presence of a specific record in input log file.

C. Event Definitions

Event rules describe how the diagnostic message will look. The message is created by calling a function *event()* from the *Tree Node* rule. In addition to the event name, the event function also accepts a packet parameter (the event is related to this packet). The idea is that from the provided packet, the time, the flow identification, and possibly other specified values are extracted and inserted into the message.

Each rule consists of a name, severity, description, instructions on how to fix the problem, and a list of fields from the provided packet. The field list contains the names according to the Wireshark terminology. An optional part of each field is its description to help the administrator understand its value. The description and suggestion may contain variables. The variables are written as $\{fieldname\}$, and will be replaced by values from the provided packet when the diagnostic report is generated.

Figure 13 shows an event describing the error that no DNS response was detected for the DNS query. From the provided packet, the queried domain name is inserted into the description and the DNS server address into the suggestion. Additional items will be also included in the output: transaction ID, queried domain name, and DNS server IP address.

```
id: reply_not_detected
severity: error
description: "No reply for query '{dns.
    qry.name}' has been detected."
suggestion: "Check if the DNS service is
     running on {ip.dst}. If yes, check
    the firewall on the server and the
    path between server and the client."
fields:
- name: dns.id
   description: Transaction ID
- name: dns.qry.name
   description: Queried domain name
- name: ip.dst
   description: Server IP address
```

Figure 13. A DNS event example, that reports that the query wasn't detected.

VI. Use Case and Evalution

The goal of the tool is on-demand diagnostic of the selected network traffic. It is essential to note that the goal is not an on-line (24/7) analysis or analysis of a large amount of data. The idea is based on a use case that when an administrator detects a problem on the network, it triggers a capture on the selected network traffic. For example, if a client with the address 192.168.0.20 is unable to establish a TLS connection with the server on the address 192.168.0.1, the administrator will capture the communication between these two stations.

We have implemented diagnostic rules for several application and service protocols. Table I shows the current list of supported protocols and their complexity in term of *Tree node* and *Fact Finder* rule count, and their capabilities in term of *Event* rule count.

Table I. Supported protocols and amount of rules and success, warning, error events which describe various protocol behavior situations.

Protocol	Tree nodes	Fact finders	Events		
			Success	Warning	Error
DHCP	24	22	10	9	4
DNS	12	12	8	4	5
FTP	24	10	15	6	7
HTTP	3	3	2	1	1
ICMP	4	2	0	0	4
IMAP	15	8	7	3	9
POP	21	7	5	10	7
SIP	38	22	15	1	8
SLAAC	8	7	1	6	1
SMB	27	25	20	3	5
SMTP	17	13	9	6	9
SSL	2	2	2	0	1
TCP	10	10	0	7	2

We have tested the functionality and performance of the implemented tool. Table II provides a sample of data from performance experiments. The execution time is divided into TShark tool processing time, time used for indexing the JSON from TShark, and the analysis time. Five files of different sizes containing some representative data are presented. The tests were performed on a CPU Intel Xeon Silver 4116 2.10 GHz and 4 GB RAM. It should be noted that only one CPU core was used for the diagnostics on the given processor, as TShark as well as the implemented tool are single-core applications.

Table II. The table shows the diagnostics execution time for the selected PCAP files of different sizes.

Size [MB]	Packets	Flows	Time [s]			
			TShark	Indexes	Analysis	Total
182.253	222 372	7155	18.717	27.407	19.678	65.802
21.569	62 471	7002	5.004	5.940	7.748	18.692
9.640	14 509	954	1.969	1.533	1.464	4.966
1.687	3 544	373	1.295	0.589	1.186	3.070
0.978	4 848	84	0.821	0.669	1.291	2.781

The output of the tool is a report in JSON format, which enables easy machine processing. We have created a web interface to visualize the report in a more human-readable format. The visualization consists of two parts. The first part shows a list of all detected events. After clicking on any event, the detail of this event is displayed in the second part. Below an event name, a suggestion for fixing the problem is displayed in which real values from the packet have replaced the message variables (written in curly brackets). After the suggestion message, the rest of the event attributes are displayed.

To demonstrate the functionality of the tool, we have diagnosed a PCAP file that was captured on a station with the IP address 10.10.1.4. The tool diagnoses all predefined protocols and displays the detected events in a hierarchical structure. Figure 14 shows this situation, with some events omitted for simplicity. For each event, a description and an icon of the event are displayed. The highest severity is then propagated from the deepest events out so that it is possible to find problem situations in a large number of events quickly. In addition to the detected error with the DNS response, it is possible to see other communications in the picture, which were without error.

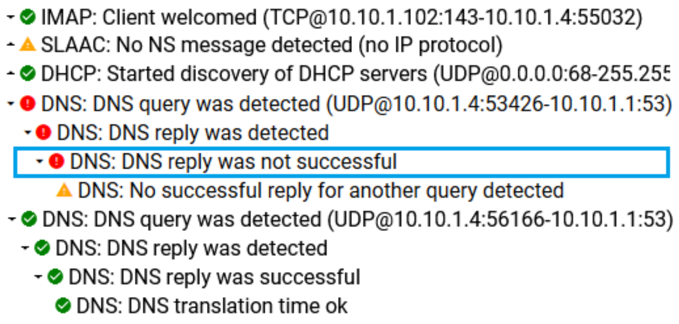
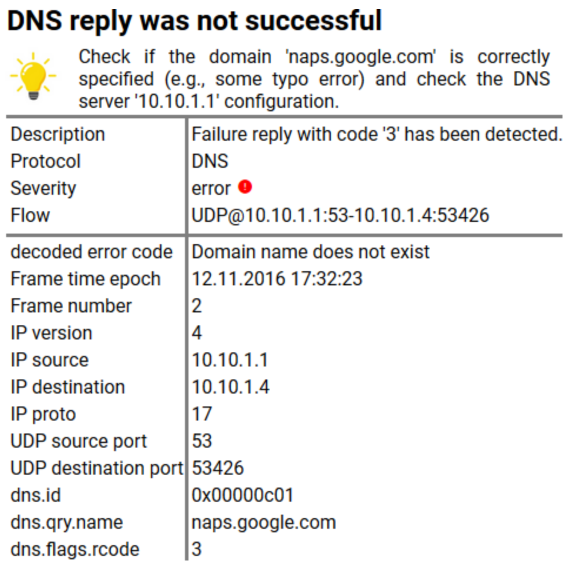

```
- ✓ IMAP: Client welcomed (TCP@10.10.1.102:143-10.10.1.4:55032)
- ⚠ SLAAC: No NS message detected (no IP protocol)
- ✓ DHCP: Started discovery of DHCP servers (UDP@0.0.0.0:68-255.255
- ● DNS: DNS query was detected (UDP@10.10.1.4:53426-10.10.1.1:53)
  - ● DNS: DNS reply was detected
    - ● DNS: DNS reply was not successful
        ⚠ DNS: No successful reply for another query detected
- ✓ DNS: DNS query was detected (UDP@10.10.1.4:56166-10.10.1.1:53)
  - ✓ DNS: DNS reply was detected
    - ✓ DNS: DNS reply was successful
        ✓ DNS: DNS translation time ok
```

Figure 14. The figure shows a list of detected events for a diagnosed PCAP file. The list contains events from multiple protocols with different severities. The screenshot was taken from the *Flowmon Packet Investigator*, a product that has integrated the proposed tool.

After selecting an event from the list (represented by a blue rectangle), a detailed description of the event is displayed. Figure 15 shows this output, which describes the reason why the domain name translation failed. In addition to the event name, the listing is divided into three sections: 1) suggestions for the administrator on how to fix the error, 2) a brief summary of the event (description, severity, and flow), and 3) attributes extracted from the packet that triggered the displayed event.


DNS reply was not successful

Check if the domain 'naps.google.com' is correctly specified (e.g., some typo error) and check the DNS server '10.10.1.1' configuration.

Description	Failure reply with code '3' has been detected.
Protocol	DNS
Severity	error ●
Flow	UDP@10.10.1.1:53-10.10.1.4:53426
decoded error code	Domain name does not exist
Frame time epoch	12.11.2016 17:32:23
Frame number	2
IP version	4
IP source	10.10.1.1
IP destination	10.10.1.4
IP proto	17
UDP source port	53
UDP destination port	53426
dns.id	0x00000c01
dns.qry.name	naps.google.com
dns.flags.rcode	3

Figure 15. The figure shows an example of diagnostic output for a DNS error. It suggests to an administrator to check the domain name and the server configuration. The screenshot was taken from the *Flowmon Packet Investigator*, a product that has integrated the proposed tool.

VII. Conclusion

Network troubleshooting can be a nightmare for administrators because of system complexity. There may not be an evident link between the issue reported by a user and the real cause of the problem. Some of the errors can be identified and analyzed by examining network traffic. However, using the traditional mostly manual approach is time-consuming and requires significant expertise. The presented paper describes the automatized approach to network traffic analysis able to identify errors using the rule-based approach. The rules encode expert knowledge and are evaluated for the captured traffic.

While the rule-based approach may be considered as an old-fashioned approach these days when the majority of research considers a machine learning-based approach, it was demonstrated that knowledge encoded in the form of rules provides an efficient method for network troubleshooting. Moreover, because of expert-designed rules, it is possible to add information that explains the possible cause of the issue and recovery options. Mainly the explainability associated with this approach seems to be the biggest benefit for users. The drawback, of course, is related to the necessity of creating and testing the rules base. Also, the method is susceptible to the quality of data sources. When the captured communication is incomplete, the method can provide incorrect results.

The performance is important for any method to be practically usable. The presented method uses a rule evaluation engine that traverses decision trees for problem domains, which can be evaluated in a reasonable timeframe. However, as nodes of the tree contain expressions that can potentially require complex operations over the input data, the set of indexes is precomputed to improve the performance.

The proof-of-concept demonstrating the approach was implemented and further finalized to the tool integrated into the commercial suite for network monitoring. The tool is commercially provided on the market by Flowmon Networks company as Flowmon Packet Investigator [26].

Future work will focus on:

- adding support of new protocols, e.g., NTP or SNMP;
- even though the current performance is good enough, it can always be better, and our goal will be to decrease the execution time of the diagnostic process;
- because the quality of the diagnostic output highly depends on the quality of the input data, we would like to create a validation technique (maybe by using machine learning techniques) to check the validity of the input data (e.g., detection of packets loss);
- after separating the processing of the input data from the Fact Finder into Data Indexer, it is now possible to create a distributive solution that consists of many data collection points across the network. At each point, the data would be indexed by the Data Indexer and sent to the central processing unit.

Acknowledgment

This work was supported by project ”Network Diagnostics from Intercepted Communication” (2017-2019), no. TH02010186, funded by the Technology Agency of the Czech Republic, the BUT FIT grant FIT-S-20-6293, ”Application of AI methods to cyber se-curity and control systems”, and by private network monitoring company Flowmon Networks.

References

[1] M. Holkovič and O. Ryšavý, "Network diagnostics using passive network monitoring and packet analysis," The Fifteenth International Conference on Networking and Services (ICNS), 2019, pp. 47–51.

[2] R. Wang, D. Wu, Y. Li, X. Yu, Z. Hui, and K. Long, "Knight's tour-based fast fault localization mechanism in mesh optical communication networks," Photonic Network Communications, vol. 23, no. 2, 2012, pp. 123–129.

[3] M. Solé, V. Muntés-Mulero, A. I. Rana, and G. Estrada, "Survey on models and techniques for root-cause analysis," arXiv preprint arXiv:1701.08546, 2017.

[4] H. Zeng, P. Kazemian, G. Varghese, and N. McKeown, "A survey on network troubleshooting," Technical Report Stanford/TR12-HPNG-061012, Stanford University, Tech. Rep., 2012.

[5] M. łgorzata Steinder and A. S. Sethi, "A survey of fault localization techniques in computer networks," Science of computer programming, vol. 53, no. 2, 2004, pp. 165–194.

[6] C. Guo et al., "Pingmesh: A large-scale system for data center network latency measurement and analysis," in ACM SIGCOMM Computer Communication Review, vol. 45, no. 4. ACM, 2015, pp. 139–152.

[7] B. Agarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker, "Netprints: Diagnosing home network misconfigurations using shared knowledge," in Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation, ser. NSDI'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 349–364.

[8] L. Lu, Z. Xu, W. Wang, and Y. Sun, "A new fault detection method for computer networks," Reliability Engineering & System Safety, vol. 114, 2013, pp. 45–51.

[9] S. Kandula et al., "Kandula, srikanth and mahajan, ratul and verkaik, patrick and agarwal, sharad and padhye, jitendra and bahl, paramvir," ACM SIGCOMM Computer Communication Review, vol. 39, no. 4, 2009, pp. 243–254.

[10] M. Luo, D. Zhang, G. Phua, L. Chen, and D. Wang, "An interactive rule based event management system for effective equipment troubleshooting," in IECON 2011-37th Annual Conference on IEEE Industrial Electronics Society. IEEE, 2011, pp. 2329–2334.

[11] A. Mohamed, "Fault detection and identification in computer networks: A soft computing approach," Ph.D. dissertation, University of Waterloo, 2010.

[12] D. Brauckhoff, X. Dimitropoulos, A. Wagner, and K. Salamatian, "Anomaly extraction in backbone networks using association rules," in Proceedings of the 9th ACM SIGCOMM conference on Internet measurement. ACM, 2009, pp. 28–34.

[13] L. Benetazzo, C. Narduzzi, P. A. Pegoraro, and R. Tittoto, "Passive measurement tool for monitoring mobile packet network performances," IEEE transactions on instrumentation and measurement, vol. 55, no. 2, 2006, pp. 449–455.

[14] K.-H. Kim, H. Nam, J.-H. Park, and H. Schulzrinne, "Mot: a collaborative network troubleshooting platform for the internet of things," in Wireless Communications and Networking Conference (WCNC), 2014 IEEE. IEEE, 2014, pp. 3438–3443.

[15] T. Qiu, Z. Ge, D. Pei, J. Wang, and J. Xu, "What happened in my network: mining network events from router syslogs," in Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010, pp. 472–484.

[16] M. Vásquez-Bermúdez, J. Hidalgo, M. del Pilar Avilés-Vera, J. Sánchez-Cercado, and C. R. Antón-Cedeño, "Analysis of a network fault detection system to support decision making," in International Conference on Technologies and Innovation. Springer, 2017, pp. 72–83.

[17] S. Jamali and M. S. Garshasbi, "Fault localization algorithm in computer networks by employing a genetic algorithm," Journal of Experimental & Theoretical Artificial Intelligence, vol. 29, no. 1, 2017, pp. 157–174.

[18] S. Kim, S. j. Ahn, J. Chung, I. Hwang, S. Kim, M. No, and S. Sin, "A rule based approach to network fault and security diagnosis with agent collaboration," in Artificial Intelligence and Simulation, T. G. Kim, Ed. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 597–606.

[19] A. De Paola, S. Fiduccia, S. Gaglio, L. Gatani, G. Lo Re, A. Pizzitola, M. Ortolani, P. Storniolo, and A. Urso, "Rule based reasoning for network management," in Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05), July 2005, pp. 25–30.

[20] C. Dong and N. Dulay, "Argumentation-based fault diagnosis for home networks," in Proceedings of the 2nd ACM SIGCOMM Workshop on Home Networks, ser. HomeNets '11. New York, NY, USA: Association for Computing Machinery, 2011, p. 37–42. [Online]. Available: https://doi.org/10.1145/2018567.2018576

[21] E. S. Ali and M. Darwish, "Diagnosing network faults using bayesian and case-based reasoning techniques," in Computer Engineering & Systems, 2007. ICCES'07. International Conference on. IEEE, 2007, pp. 145–150.

[22] B. Aggarwal, R. Bhagwan, T. Das, S. Eswaran, V. N. Padmanabhan, and G. M. Voelker, "NetPrints: Diagnosing home network misconfigurations using shared knowledge," Proceedings of the 6th USENIX symposium on Networked systems design and implementation, vol. Di, no. July, 2009, pp. 349–364. [Online]. Available: http://portal.acm.org/citation.cfm?id=1559001

[23] M. Chen, A. Zheng, J. Lloyd, M. Jordan, and E. Brewer, "Failure diagnosis using decision trees," International Conference on Autonomic Computing, 2004. Proceedings., 2004, pp. 36–43. [Online]. Available: http://ieeexplore.ieee.org/document/1301345/

[24] S. Burschka and B. Dupasquier, "Tranalyzer: Versatile high performance network traffic analyser," in 2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016, 2017.

[25] P. Casas, T. Zseby, and M. Mellia, "Big-DAMA: Big Data Analytics for Network Traffic Monitoring and Analysis," Proceedings of the 2016 Workshop on Fostering Latin-American Research in Data Communication Networks (ACM LANCOMM'16), 2016.

[26] "Flowmon products overview," https://www.flowmon.com/en/overview, accessed: 2020-May-27.

A Comparative Study of 4G and 5G Network Simulators

Christos Bouras
Computer Technology Institute and Press "Diophantus", Patras, Greece
Computer Engineering and Informatics Dept., University of Patras, Greece
email: bouras@cti.gr

Georgios Diles
Computer Engineering and Informatics Dept., University of Patras, Greece
email: diles@ceid.upatras.gr

Apostolos Gkamas
University Ecclesiastical Academy of Vella, Ioannina, Greece
e-mail: gkamas@aeavellas.gr

Andreas Zacharopoulos
Computer Engineering and Informatics Dept., University of Patras, Greece
email: st1003768@ceid.upatras.gr

Abstract— **Network simulation is a technique of utmost importance to evaluate new network performance, verify new algorithms and analyze various network topologies. It is used to find results to be expected from a hardware setup without the need for actual implementation. For this reason, there is a plethora of network simulation software applied to different scenarios to evaluate theories and hypotheses. A cellular network is an example of an extremely complex system in which different components impact the overall performance in different ways. The aim of this paper is to study the most common simulators regarding the deployment of 4G and 5G networks. In addition, this paper provides a detailed comparison of 4G and 5G simulators in order to support the academic community choose the most appropriate simulator for each project.**

Keywords- Simulator; Comparison; 4G Networks; 5G Networks; Cellular Network.

I. INTRODUCTION

The exponential increase in mobile data traffic has driven current wireless networks towards their limits and as a result, researchers should be highly motivated to create powerful next-generation mobile networks, based on the current networks trends and needs of that era. It is indicative that due to popularization of smart devices and development of Internet services, the mobile data traffic flow is expected to increase a thousand percent by the end of this decade. 4G is the fourth generation of mobile phone technology. It follows on from the existing 3G (third generation) and 2G (second generation) mobile technology. 5G will elevate the mobile network to not only interconnect people, but also interconnect and control machines, objects, and devices. The current research on 5G networks is actively moving with a high pace, meaning vendors and operators are already involved in 5G testing and trials, which is soon expected to lead to a finalized standard.

Network technology is advancing rapidly and, accompanied by expansion of network scale, have made it extremely hard to analyze network. It goes without saying that testing algorithms and protocols is extremely important since their launching in large scale is prohibitive because of uncertainty of its outcome. Network schemes can be tested either by analytical modelling or with the help of simulation tools. Although analytical modelling can indeed have very realistic results, it does not come without drawbacks, most notably the lack of precision regarding energy and memory needs and can be proven to be very expensive.

On the other hand, network simulation is used to imitate over time the operation of a real-world system enabling the observation of services and applications the network could support. It allows the researchers to model a network's behavior given the proposed changes, either with the use of mathematical formulas to calculate the interaction between the various entities of the network, or actually recording and recapping information that emerge from it. It provides the capability to manipulate most of the environment attributes to evaluate the system behavior under different circumstances and allows the comparison between alternatives to optimize network performance.

These developments render network simulators an urgent need for scientific researchers around the globe. It comes with relatively low cost and small to no risk, enabling researchers to decide and predict on network behavior with greater convenience, compared to practical networks. Therefore, there have been attempts to create diverse software for network simulation to test new algorithms and simulate network behavior. But choosing the most suitable simulator for each occasion is not always an easy decision.

In this paper, we analyze both commercial and open source state-of-the-art simulators presenting performance comparison regarding 4G and 5G networks in an attempt to provide reference to the scientific community when there is a need to choose the right software for simulation. Currently, the majority of the state-of-the-art simulation tools follow discrete event simulation methodology. This is the reason why we will only focus on this technique [1]. In [2] researchers discussed current simulators with different characteristics in different aspects. Here we study some of the most popular simulation tools that follow discrete event simulation like NS-3, OMNeT++, Riverbed and NetSim. The motivation behind this paper is to provide comprehensive review of various simulators, available for scientists allowing advanced research on 4G and 5G networks.

The rest of the paper is structured as follows. Section II presents related work and Section III provides and introduction to 4G and 5G networks. Section IV manifests importance and difficulties of simulation and Section V describes the special requirements of cellular network simulation. In Section VI the simulation tools are presented while the cumulative comparison follows in Section VII where the simulators' features and advantages are discussed. Finally, in Section VII we draw up our conclusions and we present our future work.

II. Related Work

Picking the right simulation tool is a subject that has been troubling scientists for many years. Actually, it is not the first attempt to compare simulation software, as there have been a couple published in recent years. Challenges of system-level simulation and performance evaluation and the importance of creating a stable and reliable tool for 4G and 5G in consideration of the new needs and technologies that emerge are discussed in [3]. One example is the work presented in [4] and [5], where the comparison of popular network simulators is shown.

A performance analysis, which includes open source platforms simulating a MANET routing protocol is presented in [6]. There are researches testing different routing protocols [7] in different simulators with different network parameters to evaluate the performance of network protocols. A more detailed comparison, in which in addition to open simulators, commercial platforms are also included, is presented in [8].

The issue of 4G and 5G network simulators has engaged researchers all over the world. Authors in [9] present the analysis of traffic measurements collected from commercial cellular networks in China, and demonstrate that the spatial distribution of the traffic density (the traffic load per unit area) can be approximated by the log-normal or Weibull distribution depending on time and space. After that, the authors propose a spatial traffic model, which generates large-scale spatial traffic variations by a sum of sinusoids that captures the characteristics of log-normally distributed and spatially correlated cellular traffic. The proposed model can be directly used to generate realistic spatial traffic patterns for cellular network simulations, such as performance evaluations of network planning and load balancing.

Paper [10] studies the performance trade-offs between conventional cellular and multi-hop ad-hoc wireless networks. The authors compare through simulations the performance of the two network models in terms of raw network capacity, end-to-end throughput, end-to-end delay, power consumption, per-node fairness (for throughput, delay, and power), and impact of mobility on the network performance. The simulation results show that while adhoc networks perform better in terms of throughput, delay, and power, they suffer from unfairness and poor network performance in the event of mobility. In addition, the authors present a simple hybrid wireless network model that has the combined advantages of cellular and ad-hoc wireless networks but does not suffer from the disadvantages of either.

Authors in [11] describe SimuLTE, a framework within the OMNeT++ ecosystem for simulating Long Term Evolution (LTE) networks. The main focus of SimuLTE lies on developing and testing of communication protocols and resource-allocation algorithms, with an emphasis on the impact at the system level. The paper presents two tutorials on the modeling and performance evaluation of two LTE-related research problems are presented, namely, one concerning interference coordination and one on direct-communication management. Each tutorial provides guidelines for network definition, for configuring the scenario, and their parameters. The tutorials also describe how to modify the code of the available functions. Exemplary result analysis is presented along with each tutorial, to demonstrate the evaluation capabilities of the framework.

Several simulation frameworks and tools exist to deal with these constraints of scalability and time. However, all of them require profound background knowledge for building such a custom scenario. This crucial and necessary procedure is often time consuming and error-prone. Paper [12] presents the RACE framework which aims to solves these problems for the simulation of cellular LTE networks. RACE is intuitive and based upon real life cellular network infrastructure data as well as a realistic vehicular traffic simulation.

Paper [13] presents the Vienna 5G system level simulator, which allows to perform numerical performance evaluation of large-scale multi-tier networks, with numerous types of network nodes. The simulator is based on MATLAB and is implemented in a modular fashion, to conveniently investigate arbitrary network and parameter constellations, which can be enhanced effortlessly.

One of the main strengths of Network Simulator 3 (NS-3) is the availability of modules to simulate cellular networks, including LTE and mmWave/NR deployments. These implementations model the protocol stack with a high level of detail, and, thanks to the integration with the whole NS-3 code base, make it possible to study end-to-end scenarios and identify complex interactions between the different components of a network. Authors in [14] discuss the current limitations of this platform and suggest directions for future work that could improve the accuracy of the simulations.

Carrier aggregation (CA) technology was introduced in 3GPP specification in Release 10, in 2011, as part of Long-Term Evolution Advanced (LTE-A) standardization. Authors in [15] describe the CA extension of the LTE module of NS-

3 network simulator. This paper provides description of example scenarios and the validation of the carrier aggregation feature providing a performance comparison of the LTE system with and without CA capability.

LTE/Wi-Fi Link Aggregation (LWA) and LTE WLAN Radio Level Integration with IPSec Tunnel (LWIP) are two approaches put forward by the 3rd Generation Partnership Project (3GPP) to enable flexible, general, and scalable LTE-WLAN interworking in the context of 5G. These techniques enable operator-controlled access of licensed and unlicensed spectrum and allow transparent access of operator's evolved core. Paper [16] describes the design details of LWA and LWIP protocols and presents the first NS-3 LWA and LWIP implementations in NS-3.

III. 4G Networks and 5G Networks

4G is the fourth generation of mobile phone technology. It follows on from the existing 3G (third generation) and 2G (second generation) mobile technology. 2G technology launched in the 1990s and was capable of making digital phone calls and sending texts. Then 3G came along in 2003 and made it possible to browse web pages, make video calls and download music and video on the move. 4G technology builds upon what 3G offers but does everything at a much faster speed.

The benefits of 4G fall firmly into three categories. These are:

- Improved download/upload speeds.
- Reduced latency.
- Crystal clear voice calls.

Standard 4G (or 4G LTE) is around five to seven times faster than 3G, offering theoretical speeds of up to around 150Mbps. That equates to maximum potential speeds of around 80Mbps in the real world. With standard 4G someone can download a 2GB HD film in 3 minutes 20 seconds on a standard 4G mobile network, while it would take over 25 minutes on a standard 3G network.

Meanwhile, researchers and vendors expressed a growing interest in 4G wireless networks that support global roaming across multiple wireless and mobile networks—for example, from a cellular network to a satellite-based network to a high-bandwidth wireless LAN. With this feature, users have access to different services, increased coverage, the convenience of a single device, one bill with reduced total access cost, and more reliable wireless access even with the failure or loss of one or more networks. 4G networks also feature IP interoperability for seamless mobile Internet access and bit rates of 50 Mbps or more.

5G will elevate the mobile network to not only interconnect people, but also interconnect and control machines, objects, and devices. It will deliver new levels of performance and efficiency that will empower new user experiences and connect new industries. 5G will deliver multi-Gbps peak rates, ultra-low latency, massive capacity, and more uniform user experience.

5G is a new kind of network: a platform for innovations that will not only enhances today's mobile broadband services but will also expand mobile networks to support a vast diversity of devices and services and connect new industries with improved performance, efficiency, and cost. 5G will redefine a broad range of industries with connected services from retail to education, transportation to entertainment, and everything in between. We see 5G as technology as transformative as the automobile and electricity.

In general, 5G use cases can be broadly categorized into three main types of connected services:

- Enhanced Mobile Broadband: 5G will usher in new immersive experiences, such as VR and AR, with faster, more uniform data rates, lower latency, and cost-per-bit.
- Mission-Critical communications: 5G will enable new services that can transform industries with ultra-reliable/available, low latency links—such as remote control of critical infrastructure, vehicles, and medical procedures.
- Massive Internet of Things: 5G will seamlessly connect a massive number of embedded sensors in virtually everything through the ability to scale down in data rates, power and mobility to provide extremely lean/low-cost solutions.
- A defining capability of 5G is also the design for forward compatibility—the ability to flexibly support future services that are unknown today.

IV. Network Simulation

Creating the desirable network in a real time scenario is challenging as researchers' needs and requirements may vary depending on the situation. For that reason, there is a great number of software that can be used in every case. In any case, one feature is certain and non-negotiable, all of the simulation software has to enable a user to represent a network topology, specify the nodes on the network, and of course the links and the traffic between them. Of course, there may be simulators of much higher complexity that permit specification of every detail regarding the protocols they wish to use for handling traffic in a network laying quite solid foundations for future real time implementation. Simulators may come with text-based applications that can provide a not very intuitive interface, which could nevertheless allow evolved tools for customizing or with graphical applications capable of granting users an easy and fast way to visualization of the workings of the environment they wish to examine.

The simulation of wireless networks is even more complicated due to the nature of wireless networks. The basic concept of network simulation can be found in Fig. 1.

Differentiating simulators is most commonly based in terms of speed, accuracy, cost and convenience of use. The majority of the simulators provide a multi-protocol and modularity framework. There are some network simulators in companies that are developed exclusively for business, while others are developed by research institutes and/or universities to be used for researching purposes. In general, commercial software is not open, more expensive but can provide more protocol and model support while the other simulators are free but may not be as applicable.

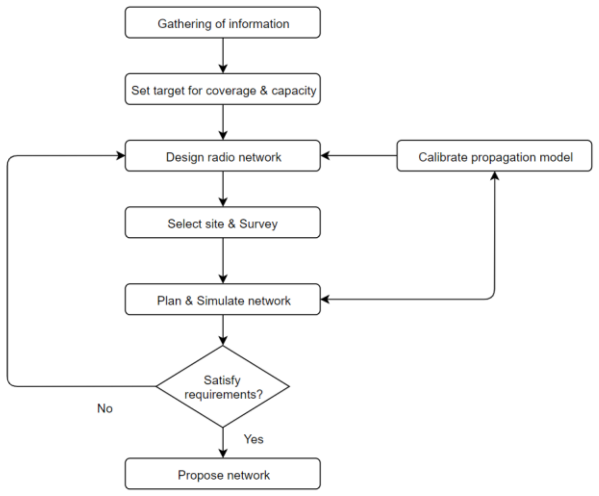


Figure 1. Network simulation

The criteria based on which the different types of simulators will be judged regard system performance, ease of learning and ease of use, the presence of Graphical Interface support, availability of the tool etc. There is general information as well as properties of the software. They can be found gathered in Table I in Section VII.

V. Cellular Network Simulation

A cellular network is an example of an extremely complex system in which different components impact the overall performance in different ways. For example, user mobility, data traffic, and the propagation model can affect the network behavior.

According to [12] cellular network simulator provides multiple simulation modules, each focusing on the modeling of a different modules, which can be interfaced together to obtain a thorough description of the overall system. The simulation of cellular networks involves both the modules implementing the cellular technology, the core network, and other modules, which take into account and influence the system behavior. More specifically these modules include the following:

- Channel Model: General speaking channel model describes the propagation of the signal through the surrounding environment, taking into account different parameters like carrier frequency, the presence of blocking objects, etc. Given that the performance of cellular networks strongly depends on the nature of the wireless channel, proper modeling of its behavior is of primary importance to obtain accurate simulation results.
- Application Model: The application model is utilized to simulate the client traffic. The use of unrealistic traffic models may prompt assessing the simulation under conditions that are not representative of those encountered in real-world scenarios, something that may lead to wrong results.
- Mobility Model: The mobility model characterizes the movements of mobile users in the environment. The usage of proper mobility models is an important aspect to consider in order to obtain accurate results.

VI. Simulators And Their Features

The following section presents the main simulators studied, their main properties, the major strengths and most important weaknesses. As mentioned above, the software in question follow discrete-event simulation. This methodology means that the operation of the system is modeled as a discrete sequence of events in time and its behavior can be simulated by modeling the events in the system where user has to set the scenarios in the right order. Also, they are chosen due to their popularity and widespread use.

A. NS-3

The NS-3 is a discrete-event network simulator developed mainly to be used for research and educational purposes. Based on the development on NS-2, the NS-3 project was launched in 2006 and is licensed under the GNU GPLv2 license and is applicable for development and research for free. It should be noted that although NS-3 was based on NS-2, it is to not be mistaken as an updated version of it, rather than as an attempt to replace it, meaning that NS-3 does not provide backward compatibility with NS-2. It defines a model of working procedure of packet data networks and provides an engine for simulation. Without deviating from its predecessor and base, NS-3 uses two key languages in C++ and Python. While the simulator is developed exclusively in C++ with optional python bindings, this allows the users the freedom to choose between C++ and Python for the scripts of simulation they write. It should be noted that in any case, both languages work very effectively on NS-3. The specified software also provides Graphical Interface for the results' visual presentation, with the use of animators. Finally, NS-3 comes with a powerful library enabling the users to do have the desired outcome, allowing them to edit NS-3 itself.

The main features of Network Simulator 3, which also differentiate it from NS-2 include:

1. Different software core: NS-3 has its core written entirely in C++ and with Python scripting interface [17].
2. Virtualization support: Implements the use of lightweight Virtual Machines.
3. Software integration: allow the inclusion of more open-source networking software, which means that the simulation models do not have to be rewritten.
4. Attention to realism: real computers are emulated in more detail by protocol entities.

Due to its features, NS-3 displays several strengths, such as:

- High modularity.
- A lot more flexibility in comparison to most simulation software.

- Easier and more credible model validation via ported code support.
- Enable simulation for a plethora of protocols.
- Wide range of use for expanding or enhancing existing networks.
- Allows Software integration.

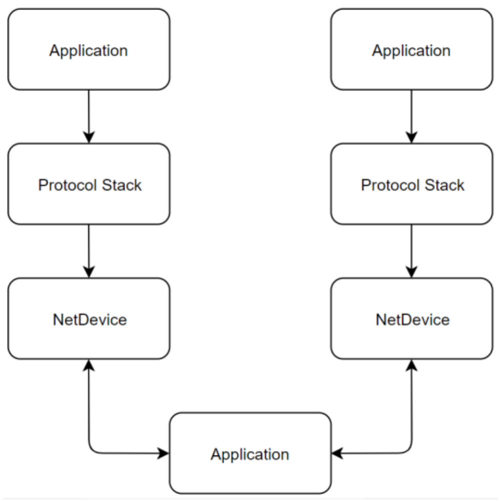


Figure 2. Architecture of NS-3

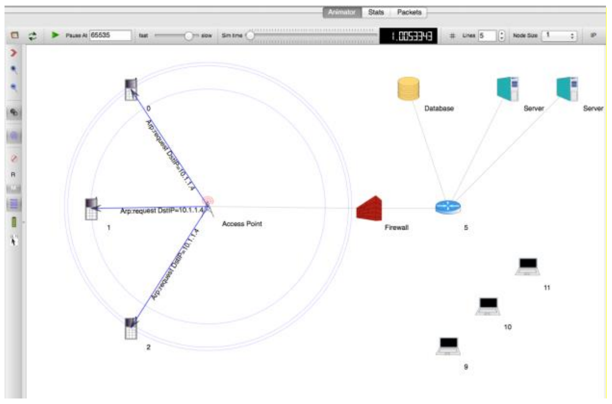


Figure 3. An example of NetAnim

However, it also has some weaknesses:

- NS-3 still suffers from lack of credibility.
- NS-3 attempts to replicate the successful approach of NS-2 but the latter was used by many organizations that contributed by adding to models and components [8].
- There is an imperative need of active maintainers who will respond to the user questions, write adequate documentation, fix reported bugs, and ensure the correct service of the system.
- The aforementioned maintainers are also needed in order to have financial advantage of NS-3 like other commercially released simulators.

The basic structure of NS-3 can be found in Fig. 2. In Fig. 3, an example of NetAnim is shown, a software executable that allows display of topology and animation of packet flow [17].

NS-3 provides support for 4G networks through the LTE module, which consist of two main components:

- The LTE Model. This model includes the LTE Radio Protocol stack.
- The EPC Model. This model includes core network interfaces, protocols and entities.

NS-3 supports 5G networks simulations through "mmWave Cellular Network Simulator module" [18]. This module includes a number of detailed statistical channel models as well as the ability to incorporate real measurements or raytracing data.

B. *OMNeT++*

Publicly available since 1997, OMNeT++ [19] is an extensible, modular, discrete event simulation software [20]. Although it can successfully model complex IT systems, multiprocessors, distributed hardware architectures, it is more often used for computer networks simulation, both wireless and wired. It is written thoroughly in C++. Using the software under the Academic Public License makes it free for non-benefit or academic use. Its free disposal combined with the tool's extensibility and the amount of available online documentations have made it very popular in the academic community. The motivation behind the development of OMNeT++ is to bridge the gap between research-oriented, free simulators like NS-3 and commercial alternatives like Riverbed that are much more high-priced. It is a component-based architecture and components (called modules) are programmed entirely in C++. They are then assembled into larger components and models with the use of NED, a language of higher level. Its modular architecture allows the simulation kernel to be easily embedded into almost every application.

The software has great GUI support and the simulation environment also offers a compiler for the NED topology description language (NEDC), graphical network editor for NED files (GNED), GUI for simulation execution (Tkenv), command-line user interface for simulation execution (Cmdenv) [19] [21].

The most important feature of the simulator is that the modules are assembled by reusable components to be combined in different ways. Another important feature is that OMNet++ is basically a framework approach, providing the groundwork to develop various simulations models to meet different application areas requirements, which subsequently follow their release cycles. Currently, it is on version 5.4.1.

The simulator's strengths can be summarized as follows:

- Makes it easier to trace and debug.
- Can be used to model most hardware with accuracy.
- It offers wide GUI support via a complete, robust environment.
- Provides Reusable modules that can be combined in different ways.

While its weaknesses include:

- User still have to do a pretty important amount of background work due to the lack of variety of protocols offered and implemented.
- The mobility extension can be found somewhat incomplete.
- It offers poor analysis and management of typical performance.

The structure of OMNeT++ simulation system can be found in Fig. 4 and an example of simulation in Fig. 5.

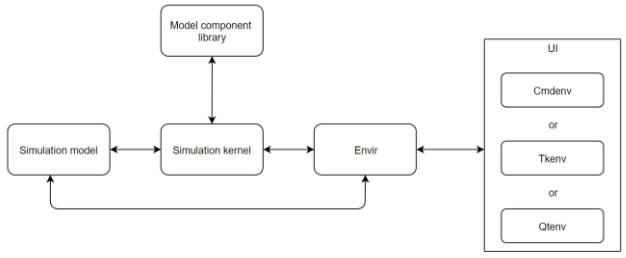


Figure 4. Structure of OMNeT++ simulator

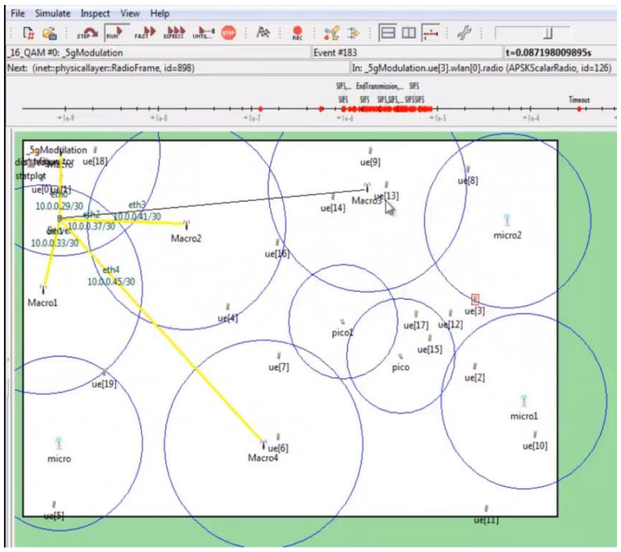

File Simulate Inspect View Help
_16_QAM #0: _5gModulation
Event #183
t=0.087198009895s
Next: (inet::physicallayer::RadioFrame, id=898)
In: _5gModulation.ue[3].wlan[0].radio (APSKScalarRadio, id=126)
Macro1
Macro2
Macro3
Macro4
micro
micro1
micro2
pico
pico1

Figure 5. Example of simulation on OMNeT++

OMNet++ provides support for 4G network simulation through SimuLTE module [22]. SimuLTE is an innovative simulation tool enabling complex system level performance-evaluation of LTE and LTE Advanced networks (3GPP Release 8 and beyond) for the OMNeT++ framework. SimuLTE is written in C++ and is fully customizable with a simple pluggable interface. One can also develop new modules implementing new algorithms and protocols. SimuLTE is an open source project building on top of OMNeT++ and INET Framework [23].

C. Riverbed Modeler

OPNET (Optimized Network Engineering Tools) Modeler is the development environment of OPNET simulator and is targeted for both research purposes and development. It was one of the most popular commercial simulation software by the end of 2008 and being in the market for such a long period, it managed to occupy a large share of it. Nowadays it is part of Riverbed Modeler. Its flexibility allows it to be highly useful in studying communication applications, protocols and networks. It offers the users vast and impressive visual interface, due to its commercial nature. Using the graphical editor interface, the users are able to build whole network topology and entities from the application layer all the way to the physical layer and the mapping from the graphical design to the implementation of the real systems is created using Object-Oriented programming. All topologies' configuration and simulation results can be presented very intuitively and visually. The users also have the freedom to adjust the parameters and quickly repeat experiments using the graphical interface, performing tests for various scenarios [21]. Riverbed is based on a mechanism called discrete event system.

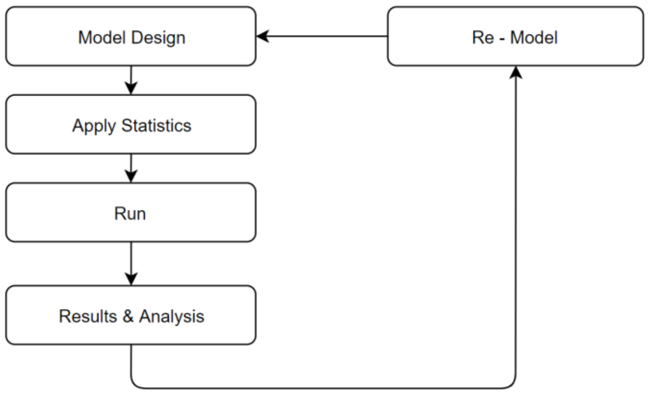


Figure 6. Simulation Workflow of Riverbed Modeler

According to the authors of [2] OPNET can be used to carry through with three functions:

1. modeling: it provides a vary intuitive and visually rich GUI, allowing users to develop a great variety of models.
2. simulating: It uses three different technologies.
3. analysis: the results originating from the simulation process can be presented and analyzed using the simulators tools, such as user-friendly charts, animations or statistics.

Important features of the Riverbed system are that the organization of the networks is accomplished via hierarchical structure plus the fact that graphical interface and programming tools are available to users to define protocols or packet format.

Some strengths of the system include:

- Fast discrete event analytical simulation engine [8].
- Reduces simulation runtime by utilizing parallel and distributed capabilities [24].
- Allows quick correlation of graphical result with network behavior and easy interpretation.

While some weaknesses could be:

- It only supports a small number of nodes within a single device.
- Simulation is inadequate in case there are long periods where nothing happens.
- Provided GUI might be powerful but its use it rather complicated.
- Sampling resolution sets the limit for the result accuracy.

The simulation workflow of Riverbed modeler can be found in Fig. 6 and the Graphical Interface in Fig. 7.

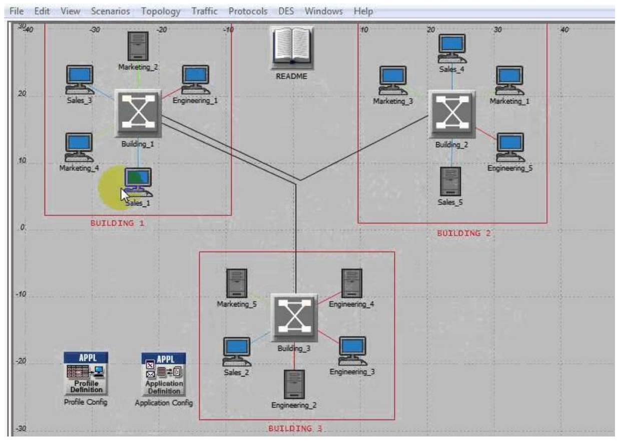


Figure 7. GUI of Riverbed Modeler

OPNET provides native support of LTE network simulation. Authors of [25] introduce modelling, simulation and Electromagnetic Compatibility (EMC) analysis methods based on OPNET for IMT-2020(5G) systems. The models presented on this paper can be used for network planning, design and performance analysis in the following research. This paper shows the new method of EMC analysis and new application of network simulation technology.

D. NetSim

NetSim is a stochastic discrete event simulator targeted for experimentation and research on networks. It is a leading network simulation software for protocol modelling and simulation, allowing us to analyze computer networks with unmatched depth, power and flexibility [26]. It is developed in 1997 by Tetcos. Its native development environment acts as the interface between User's code and NetSim's protocol libraries and simulation kernel [27]. NetSim is available as Pro, Standard or Academic versions and is built on a common design framework of high-level architecture and code. Every version has of course different features, supports different options and has a different price. NetSim is more versatile than most of the other software and robust with an excellent and easy to use graphical interface. It should be noted that it is capable to provide performance metrics at abstraction levels from network to node and creates a packet trace with all of the necessary details. Its main limitation is that it is follows a single process discrete event simulation methodology. This means that it uses a single event queue for the needs of the simulation and at any given time, it contains one entry for each station on the network. Currently it is on Version 10.

The major benefits are (a) programmability, (b) architectural accuracy, and (c) flexibility.

NetSim's strengths include:

- It offers a powerful, user friendly GUI that makes its use rather simple.
- Allows data packet flow visualization using its built-in animator.
- Users can extract performance analysis metrics in various levels.
- Its analysis framework offers various graphical options and enables intra and inter-protocol performance comparison.
- Some weaknesses could be identified:
- All of the versions are commercial, meaning there is no free way of usage.
- It is a single process discrete event simulator.

The graphical interface of Netsim can be found in Fig. 8.

NetSim provides support for 4G network simulation through LTE / LTE – Advanced module [28]. NetSim Long-Term Evolution (LTE) Model Library provides high fidelity simulation of 4G / 4.5G cellular networks based on the 3GPP TS 36.xxx standards. It includes models of nodes called MME (Mobility Management Entity), eNodeB (Base Station), Relay and UE (Mobile Station) and each has detailed MAC and PHY models.

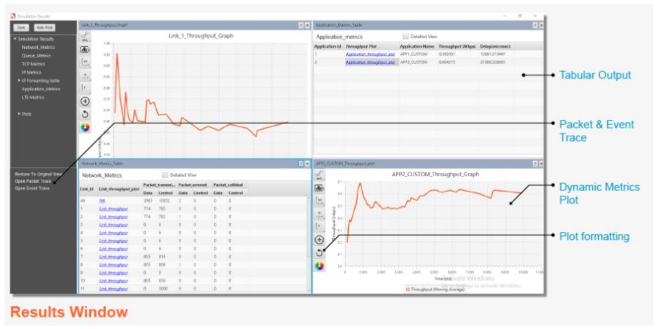


Figure 8. GUI of Netsim

During 2019, NetSim starts to provide support of 5G Network (beta software) through 5G NR mmWave module [29]. This module is in beta stage and provides the following features:

- Edge-to-edge and Edge-to-core simulation covering.
- GUI based with Drag and Drop, Packet Animator and Results Dashboard.
- 5G library interfaces with NetSim's proprietary TCP/IP stack providing simulation capability across all layers of the network stack.
- Discrete Event Simulation (DES) with event level debugging to inspect and control the simulation.
- Application Models - FTP, HTTP, Voice, Video, Email, DB, Custom.
- Packet level simulation with detailed packet trace.

VII. Cumulative Comparison

The simulation comparison is shown in Table I, where the criteria are presented and whether they are fulfilled. The comparison was an overall comparison and it is based on both general information as well as properties of the software.

The general information that can be found on the upper section of the table, e.g., supported language & OS, license type, GUI support and technical properties of the software are compiled under the middle rule, e.g., simulation event type, scalability and network visualization tool.

All the simulators studied in this paper support tools that help the visualization of the network. They also allow scenarios redesign and modification through parameters change and can create trace files. They offer complete documentation and are user friendly, easy to use with NS-3 proving to be the most challenging to learn. The modularity of OMNeT++ is a big advantage, although it leaves the user with quite a big amount of work to done because of the lack of protocols offered.

When it comes to communication with other simulators, Riverbed Modeler supports this feature while Omnet++, NS-3 and NetSim do not. Because of its proprietary nature, it is only natural that Riverbed can afford to simulate networks of much larger scale.

On the other hand, NS-3 is open source, OMNeT++ may not entirely be free but offers academic version for noncommercial use and NetSim offers a cheaper, alternative version for students. This means that for these versions, the simulation scale ability is more limited. Of course, the commercial versions of the latter two software, can support large scale simulations. NS-3 and OMNeT++ can be deployed in all widely used Operating Systems, contrary to Riverbed and NetSim.

As far as GUI support is compared, the graphical environment offered by all of them are found more than adequate. Of course, OMNeT++ and NetSim offer vast and powerful GUI support with many more features and abilities like analysis framework and graphical options. On the other hand, Riverbed does provide an excellent GUI but it can be judged quite complicated and not so user-friendly.

It should be noted that all the simulators are supported by a great community but, NS-3 being open source means that there are less maintainers to respond to questions or fix reported bugs and abnormalities. However, it is extremely widespread and used by many students, scientists and academics that the online community can help and offer great support. More specifically, according to [30] a Google Scholar search of the 'NS-3 simulator' results since 2017 (excluding patents and citations) yields over 2000 links (with some false positives). In addition, the IEEE digital library lists 145 NS-3 publications for 2017, and the ACM digital library lists 2579 publications matching the search term 'NS-3' in 2017. In addition, there are organized Workshops on NS-3 and the related proceedings are published in the ACM digital library [31]. The above facts ensure the important acceptance of NS-3 simulator as network research tool. In addition to NS-3, also OMNeT++ has an active community, which have organized 5 OMNeT++ Community Summits until 2018 [32]. As result, if we compare the above simulators in terms of research community support seems that NS-3 and OMNeT++ have the most active research community, which organize relative workshops about the evolution of the simulation software. This seems logical based on the fact that both NS-3 and OMNet++ can be obtained at no cost.

If we discuss about 4G networks simulations native support all the simulators in investigation provides support for 4G – LTE network simulation.

If we discuss about 5G networks simulations native support, only NS-3 and NetSim simulators supports 5G networks simulations and OMNET++, Riverbed, do not provide native support for 5G networks simulations.

TABLE I. COMPARISON OF SIMULATORS.

	NS-3	**OMNeT++**	**Riverbed**	**NetSim**
License Type	Open Source	Open Source (study & research)	Commercial	Proprietary
Language Supported	C++ & Python	C++	C & C++	C++ & Java
Supported OS	Linux, Mac OS Windows	Linux, Mac OS Windows	Linux, Windows	Windows
GUI Support	Good	Good	Excellent	Excellent
Document Available	Yes	Yes	Yes	Yes
Ease of Use	Hard	Easy	Easy	Easy
Simulation Event Type	Discrete event	Discrete event	Discrete event	Stochastic Discrete event
Available Module	Wired, Wireless Adhoc,WSN	Wired, Wireless Adhoc,WSN	Wired, Wireless Adhoc,WSN	Wired, Wireless, SN
Scalability	Limited	Enough	Large	Enough
Availability of analysis tool	Yes	Yes	Yes	No
Communication with other modules	No	No	Yes	No
Network visualization tool	Yes	Yes	Yes	Yes
Possibility to design and modify scenarios	Yes	Yes	Yes	Yes
4G native support	Yes	Yes	Yes	Yes
5G native support	Yes	No	No	No

NS-3 supports 5G networks simulations through "mmWave Cellular Network Simulator module" [33]. This module includes a number of detailed statistical channel models as well as the ability to incorporate real measurements or raytracing data. The physical and medium access control layers are modular and highly customizable. The module is interfaced with the core network of the NS-3 Long Term Evolution (LTE) module for full-stack simulations of end-to-end connectivity, and advanced architectural features, such as dual-connectivity, are also available.

During 2019, NetSim starts to provide support of 5G network (beta software) through 5G NR mmWave module [29]. This module is in beta stage and provides the following features:

- Edge-to-edge and Edge-to-core simulation covering.
- GUI based with Drag and Drop, Packet Animator and Results Dashboard.
- 5G library interfaces with NetSim's proprietary TCP/IP stack providing simulation capability across all layers of the network stack.
- Discrete Event Simulation (DES) with event level debugging to inspect and control the simulation.
- Application Models - FTP, HTTP, Voice, Video, Email, DB, Custom.
- Packet level simulation with detailed packet trace.

Especially for 5G networks simulations there are specialized simulators like NYUSIM [33]. NYUSIM is a novel channel simulation software, which can be used to generate realistic temporal and spatial channel responses to support realistic physical- and link-layer simulations and design for fifth-generation (5G) cellular communications. NYUSIM is built upon the statistical spatial channel model for broadband millimeter-wave (mmWave) wireless communication systems.

VIII. Conclusion and Future Work

Network simulation is an effective, low cost and small risk method. However, it is necessary and this why it is extensively performed by scientists in all kinds of fields to validate the research carried out. Network simulation can prove to be an essential mechanism on the hands of researchers for the analysis on network behavior and evaluation on possible network design and will remain increasingly important following the networks' growing complexity and scale. This paper contains a general overview of a number of tools used for standard network simulation, along with a comparison between them with respect to various parameters. The study confirms that picking a suitable, required and efficient simulator for the specific job of a research work can be quite demanding but bears the according results. Each simulator comes with its advantages and disadvantages and can be useful or even necessary in different cases and the choice of a fitting software should be done based on the study motive.

If we discuss about 4G networks simulations native support all the simulators in investigation provides support for 4G – LTE network simulation. If we discuss about 5G networks simulations native support, only NS-3 and NetSim simulators supports 5G networks simulations and OMNET++, Riverbed, do not provide native support for 5G networks simulations.

Our future work includes the evaluation of the investigated simulators based on the simulation of the same cellular network scenarios. The above will allows us to compare the performance of investigated simulators with parameters like time to complete simulation, accuracy of the simulation results etc.

In addition, our future work includes the comparison of the investigated simulators during the simulation of Low Power Wide Area Networks (LPWAN) like LoRA (Long Range) and NB-IoT (Narrow Band – Internet of Things). LPWAN are very important for the implementation of the Internet of Things (IoT) applications something very important because IoT is the extension of Internet connectivity into physical devices and everyday objects.

References

[1] C. Bouras, G. Diles, A. Gkamas, and A. Zacharopoulos, "Comparison of 4G and 5G Network Simulators", Fifteenth International Conference on Wireless and Mobile Communications (ICWMC 2019), 2019, pp. 13-18.

[2] J. Pan and R. Jain, "A survey of network simulation tools: Current status and future developments", Washington University in St. Louis, Tech. Rep, 2008.

[3] Y. Wang, J. Xu, and L. Jiang, "Challenges of system-level simulations and performance evaluation for 5G wireless networks", IEEE Access, vol. 2, 2014, pp. 1553–1561.

[4] V. Mishra and S. Jangale, "Analysis and comparison of different network simulators", Special Issue for International Technological Conference- 2014.

[5] X. Zhou and H. Tian, "Comparison on network simulation techniques", 17th International Conference on Parallel and Distributed Computing, Applications and Technologies (PDCAT), Dec. 2016, pp. 313–316.

[6] A. R. Khan, S. M. Bilal, and M. Othman, "A performance comparison of open source network simulators for wireless networks", IEEE International Conference on Control System, Computing and Engineering, Nov. 2012, pp. 34–38.

[7] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks", IEEE Personal Communications, vol. 8, no. 1, Feb. 2001, pp. 16–28.

[8] M. Kabir, S. Islam, M. Hossain, and S. Hossain, "Detail comparison of network simulators", International Journal of Scientific and Engineering Research, vol. 5, no. 10, Oct. 2014, pp. 16–28.

[9] D. Lee, et al., "Spatial modeling of the traffic density in cellular networks", IEEE Wireless Communications 21.1, 2014, pp. 80-88.

[10] H. Hsieh,and S. Raghupathy, "Performance comparison of cellular and multi-hop wireless networks: A quantitative study", ACM Sigmetrics Performance Evaluation Review. Vol. 29. No. 1, 2001.

[11] A. Virdis, N. Giovanni and S. Giovanni, "Cellular-Networks Simulation Using SimuLTE", Recent Advances in Network Simulation. Springer, Cham, 2019. 183-214.

[12] F. Jomrich et al., "Demo: rapid cellular network simulation framework for automotive scenarios (RACE framework)", International Conference on Networked Systems (NetSys). IEEE, 2017.

[13] M. K. Müller et al., "Flexible multi-node simulation of cellular mobile communications: the Vienna 5G System Level Simulator", EURASIP Journal on Wireless Communications and Networking 2018.1.

[14] T. Zugno et al., "Simulation of Next-generation Cellular Networks with NS-3: Open Challenges and New Directions", Workshop on Next-Generation Wireless with NS-3. ACM, 2019.

[15] B. Bojovic et al., "Towards LTE-Advanced and LTE-A Pro Network Simulations: Implementing Carrier Aggregation in LTE Module of ns-3", Workshop on NS-3. ACM, 2017.

[16] S. Afaqui et al., "Implementation of the 3GPP LTE-WLAN Inter-working Protocols in ns-3", Workshop on NS-3. ACM, 2019.

[17] https://www.nsnam.org/, "NS-3 network simulator", [Online; accessed 27-July-2018].

[18] M. Mezzavilla, M. Zhang, M. Polese, R. Ford, S. Dutta, S. Rangan, and M. Zorzi, "End-to-end simulation of 5g mmwave networks", IEEE Communications Surveys Tutorials, vol. 20, no. 3, thirdquarter 2018, pp. 2237–2263.

[19] https://www.omnetpp.org/, "Omnet++ official site", [Online; accessed 27-July-2018].

[20] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment", 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops, page 60. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, 2008.

[21] G. Borboruah and G. Nandi, "A study on large scale network simulators", International Journal of Computer Science and Information Technologies, vol. 5, no. 6, 2014, pp. 7318–7322.

[22] https://simulte.com/, "LTE User Plane Simulation Model for INET & OMNeT++", [Online: accessed 24-December-2019].

[23] https://inet.omnetpp.org/, "INET Framework An open-source OMNeT++ model suite for wired, wireless and mobile networks", [Online: accessed 24-December-2019].

[24] https://www.riverbed.com/gb/products/steelcentral/steelcentral-riverbed-modeler.html, "Riverbed modeler" [Online; accessed 9-May-2020].

[25] J. Chenet et al., "Modeling and simulation of IMT-2020 (5G) systems and satellite communication systems based on OPNET network simulation technology", 3rd IEEE International Conference on Computer and Communications (ICCC). IEEE, 2017.

[26] https://www.tetcos.com/, "Netsim official developer" [Online; accessed 9-August-2018].

[27] S. Siraj, A. Gupta, and R. Badgujar, "Network simulation tools survey", International Journal of Advanced Research in Computer and Communication Engineering, vol. 1, no. 4, 2012, pp. 199–206.

[28] https://www.tetcos.com/lte.html, "LTE / LTE – Advanced", [Online: accessed 24-December-2019].

[29] https://www.tetcos.com/5g.html, "5G NR mmWave", [Online: accessed 24-December-2019].

[30] https://www.nsnam.org/about/statistics/, "Statistics 2018", [Online: accessed 24-December-2019].

[31] https://www.nsnam.org/research/wns3/, "Workshop on ns-3", [Online: accessed 24-December-2019].

[32] https://summit.omnetpp.org/archive/2018, "Omnet++ community summit 2018", [Online: accessed 24-December-2019].

[33] S. Sun, G. Maccartney, and T. Rappaport, "A novel millimeter-wave channel simulator and applications for 5G wireless communications", IEEE International Conference on Communications, ICC 2017.

Energy-Aware Technology Comparisons for 5G Mobile Fronthaul Networks

Line M. P. Larsen, Michael S. Berger, Henrik L. Christiansen
Department of Photonics Engineering
Technical University of Denmark, DTU
Kgs. Lyngby, Denmark
e-mail: lmph@fotonik.dtu.dk, msbe@fotonik.dtu.dk, hlch@fotonik.dtu.dk

***Abstract*— Communication networks are not only important to society, they also consume a lot of energy. Recent years research has focused on Cloud-Radio Access Network (C-RAN) to decrease the energy consumption of mobile networks. Hence, this work investigates how to lower the energy consumption in the fronthaul network by choosing the right C-RAN functional split and the right type of fronthaul network. Different functional splits assign different loads to the fronthaul network, and this work considers how much impact the data load has on the fronthaul network's energy consumption. At the same time different types of fronthaul network impact the energy consumption in different ways. The paper provides models to be used for calculating the energy consumption in different network types and for the choice of different functional splits. Results show huge differences in energy consumption between especially wired and wireless technologies. Considering the same functional split, the difference between the most energy consuming transport technology and the least energy consuming is 99.5% for LTE and 99.6% for 5G. For the Ethernet overlay alone, the difference in energy consumption between the, with regard to energy consumption, best and worst case scenario is 99.3% per switch.**

Keywords- Energy consumption; green networking; Ethernet; fronthaul; C-RAN; functional split; 5G; PON; PtP; CWDM.

I. Introduction

This paper is an enhanced version of [1]. The Information and Communications Technology (ICT) sector counts for over 2% of the world's carbon emissions nowadays [2]. However, the energy consumption of the ICT sector is forecasted to increase by 8% by 2030 in the best case scenario, and by 20% in the worst case scenario [2]. The ICT sector covers many areas and one of them being mobile networks. Mobile networks are growing the most, among all ICT sectors, in terms of number of subscribers, traffic demand, connected devices and offered services [3]. The trend in mobile networks is that more and more capacity is required and the coverage should be everywhere. Hence, base stations are widely deployed to cover the largest area possible, in order to satisfy the users' needs. The next generation of mobile networks, the 5th Generation (5G), is approaching and promises more capacity and higher bitrates. Thus, an important parameter to consider is how this growth will affect the energy consumption in mobile networks.

In the mobile network's base stations, the power amplifier takes up most of the energy consumption, next comes the baseband processing and then the cooling [3]. Cloud-Radio Access Network (C-RAN) architectures have been introduced to lower these parameters. In C-RAN, the radio frequency and

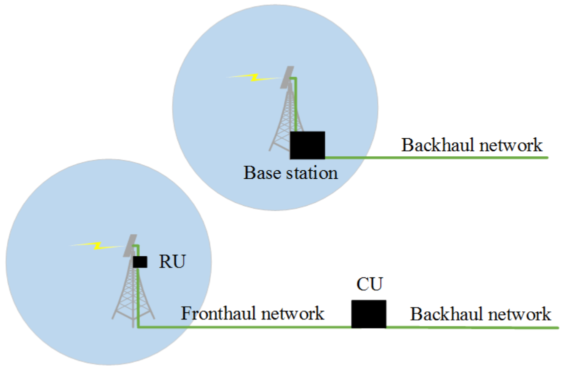


Figure 1. Comparison of traditional base station and C-RAN.

baseband processing functions from the base station are split in two units referred to as the Radio Unit (RU) and the Centralized Unit (CU). The concept is illustrated in Fig. 1. The RU is located close to the antenna at the antenna mast, thereby it is convection cooled and settles for a smaller amplifier. The CUs from several cells can be gathered in a datacenter, where it is possible for them to share processing powers when not used at the same time. Hence, C-RAN will have the possibility of saving energy consumption in the three most energy consuming parameters of the traditional base station. The RU and the CU are connected by a network segment called the Fronthaul (FH) network [4]. Originally, only the radio frequency functions were present in the RU, as the FH network required very large bitrates in order to transport a constant stream of raw In-phase and Quadrature (IQ) data blocks. These blocks of raw IQ data were transported using a special protocol, for example Common Public Radio Interface (CPRI). Recently, the concept of Functional Splits (FS) has been scrutinized, leaving more processing functions in the RU. The more functions are left locally in the RU, the lower the bitrate on the FH network, and gives the possibility of a bitrate varying with user load, but also a larger and more complex RU. Additional information regarding the FSs can be found in [4], which provides an in-depth analysis of the FSs including latency and impact on FH network. Selected FS options are illustrated in Fig. 2. Fig. 2 shows the RU and CU separated by the FH network, which is illustrated by a green dotted line. To the right in Fig. 2, the Long Term Evolution (LTE)/LTE-Advanced(LTE-A)/5G protocol stack illustrates the location of the different FSs selected for this paper. The LTE/LTE-A/5G protocol stack consists of, from the bottom up: the Radio Frequency functions (RF), the physical processing (PHY), the Media Access Control (MAC), the Radio Link Control (RLC) and the Packet Data Convergence Protocol (PDCP). Further description of the protocol stack layers can be found in [4]. On the right side of the figure is a table stating the FH bitrates for LTE, LTE-A and 5G considering different FSs. These FH bitrates are based on calculations in [5] and extended using the parameters stated in Table I, to also include LTE and 5G. The FH bitrates are only considered for the Downlink (DL) direction. The FH bitrates reveal how extremely capacity demanding the FH is, it puts large requirements to the network carrying it.

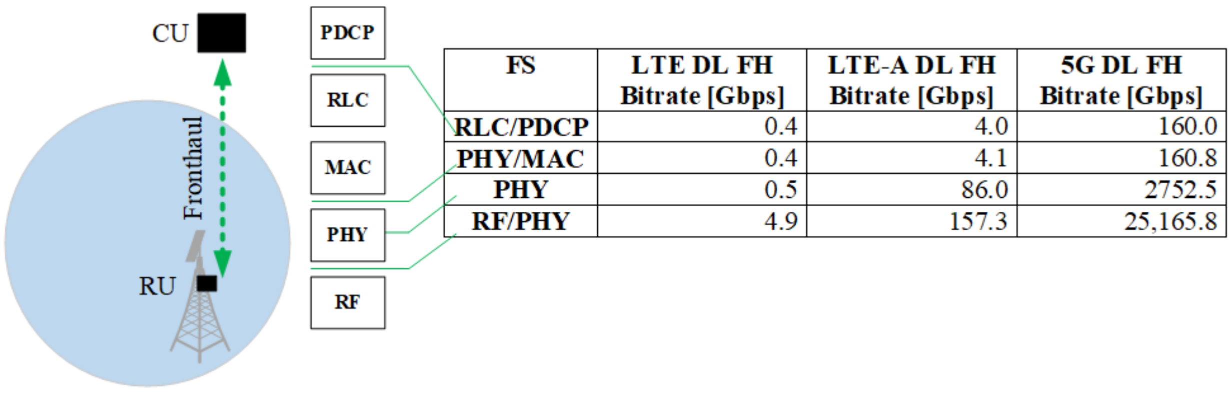
This work investigates how different FSs combined with different FH network types, impact the energy consumption in the FH network. The contribution of this paper includes new perspectives on energy consumption for different FH network options. The different FH network types considered in this paper are: point-to-point (PtP) fiber, Microwave Radio (MWR), Passive Optical Networks (PON) and Coarse Wavelength Division Multiplexing (CWDM) networks, all of these are compared with an over layer of Ethernet. These FH network types are chosen based on the overview in [6], where the physical layer technologies that can carry an overlay of Ethernet are chosen. This paper is organized as follows: Section II provides an overview of research in this field. Section III introduces FH networks in general. Section IV presents energy consumption in the physical layer including: PtP fiber, MWR, CWDM and PON FH networking. Section V presents Ethernet FH networking including a detailed model for energy consumption in the Ethernet FH network. In Section VI, the different layers energy consumptions are combined. Section VII discusses the results provided, considering how to obtain an energy efficient FH network for 5G. Section VIII concludes the paper. Compared to [1], then this work differentiates by introduction of Sections IV and VI. Remaining sections have been updated and modified.

TABLE I. PROPERTIES FOR FRONTHAUL BITRATE CALCULATIONS.

	RATs		
	LTE	*LTE-A*	*5G*
Bandwidth	20 MHz	100 MHZ	400 MHz
# Antennas	2	32	256
# Spatial layers	2	8	12
Modulation order	16 QAM	256 QAM	256 QAM
Sample rate	30,72 MHz	30,72 MHz	614,4 MHz
# Subcarriers	1200	6000	24000
# Resource element blocks	100	500	2000

II. STATE OF THE ART

C-RAN has been the topic of much research in recent years. A detailed description of the technology is found in [7]. Liu et al. state in [8] how PtP connections refer to the classical FH transporting IQ data blocks over what is now known as FS 8. Recent years research has looked into new FH opportunities, evolving the FH from PtP to shared connections. The Next Generation Mobile Networks (NGNM) provides in [6] a comprehensive study of the different options for FH transport including the five technologies selected for this paper: Ethernet, PtP, MWR, PON and CWDM networks. FH networks have been the topic of some energy efficiency investigations: Fathy et al. [9] present a power model for a RF/PHY split PON FH considering sleep mode and active RUs. They find that the average network power consumption is lower using their "greedy selection" algorithm. The work in [10] investigates the energy consumption in the RU considering different FSs, digital and analogue. In [11] Tan et al. analyze the energy consumption in RF/PHY split stating that 90 % of the energy is consumed by the RU, 9% by the CU datacenter and 1% by a 10G Ethernet PON FH network. The work in [12] by Kondepu et al. investigates the energy efficiency for the FH



FS	LTE DL FH Bitrate [Gbps]	LTE-A DL FH Bitrate [Gbps]	5G DL FH Bitrate [Gbps]
RLC/PDCP	0.4	4.0	160.0
PHY/MAC	0.4	4.1	160.8
PHY	0.5	86.0	2752.5
RF/PHY	4.9	157.3	25,165.8

Figure 2. The FSs considered and their corresponding FH bitrates for LTE, LTE-A, 5G. The FSs are illustrated in the LTE protocol stack with upper layer PDCP and lowest RF.

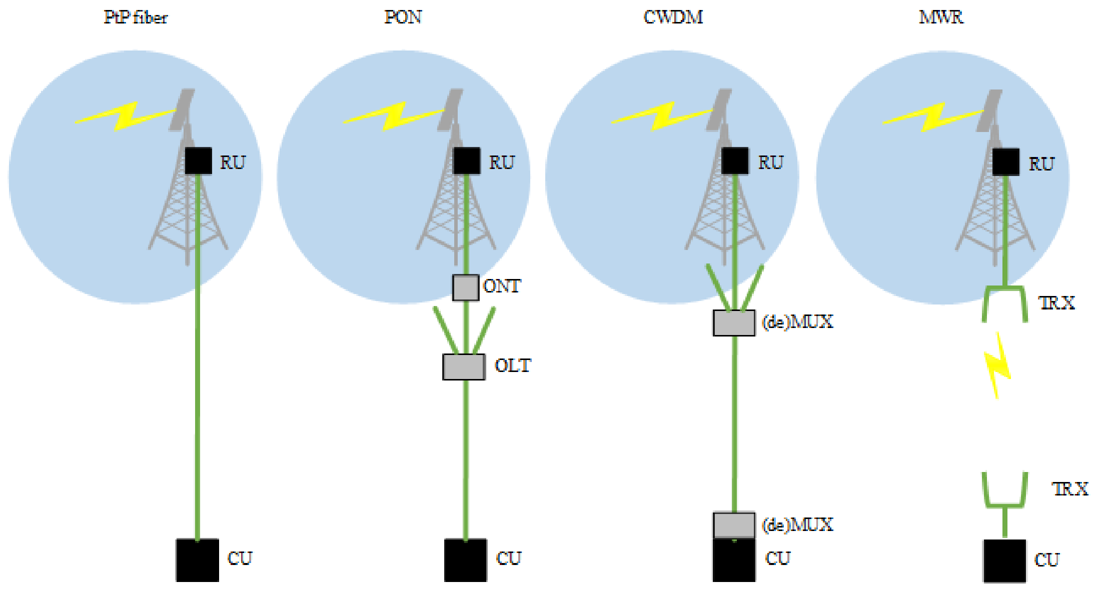


Figure 3. Illustration of different solutions for FH transport technologies: PtP, PON, CWDM and MWR

network for a flexible FS, by switching on and off resources using Software Defined Networking (SDN).

With regard to the findings provided in this section, this work represents an uninvestigated area of looking into the FH energy consumption of multiple FH options while considering different FSs individually.

III. Fronthaul Networks

FH transport can use many different types of technologies, wired and wireless. The FH network consists of different elements, depending on the type of network. Networks operate in layers, and technologies must be evaluated based on, which layer they operate. The network types considered for the physical layer in this work covers fiber PtP, MWR, PON and CWDM. On top of this, in the data link layer, this work considers Ethernet. The layered transmission considered in this work is illustrated in Fig. 4.

The figure illustrates how the transmission paths are determined by Ethernet but the physical transmission, or moving of data is performed in the physical layer.

In the future 5G network, the RAN will be expanded with more antennas. This will increase the demands to the FH network even more, as not only higher bitrates shall be transported, but also more streams are present from the higher numbers of RUs and antennas. Our model (1) estimates the amount of RUs in an area covered by one CU assuming a circular coverage area. Equation (1) takes the area of the coverage area and divides it by the area of each cell, then sums up for all RATs:

$$N_{RU} = \sum_{RAT} \frac{\pi \cdot D_{MAX}{}^2}{\pi \cdot RAT_n{}^2}, N_{RU} \in \mathbb{N}_0 \qquad (1)$$

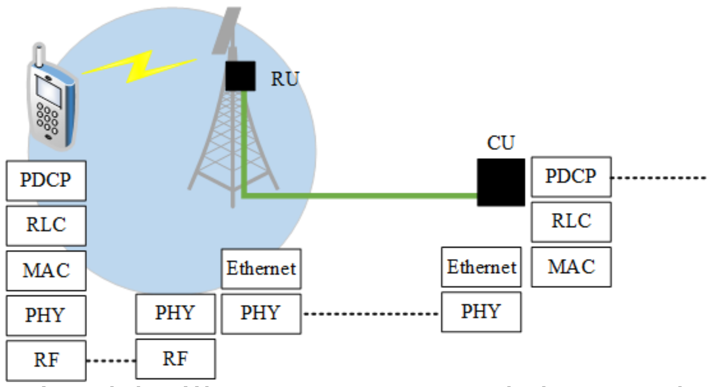


Figure 4. FH transmission over PHY layer, which could be PtP, PON, CWDM or MWR with Ethernet on top. The FS illustrated in this figure is PHY/MAC.

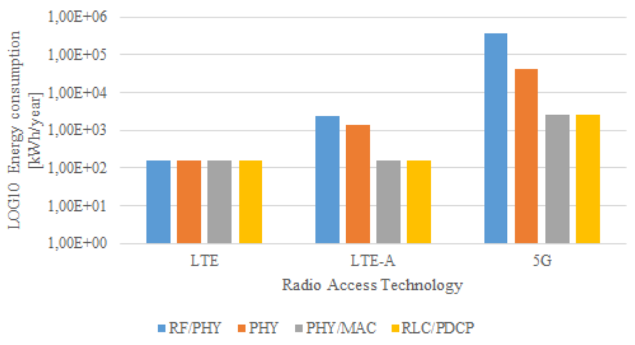


Figure 5. Energy consumption by RAT for one RU-CU connection using PtP/CWDM FH.

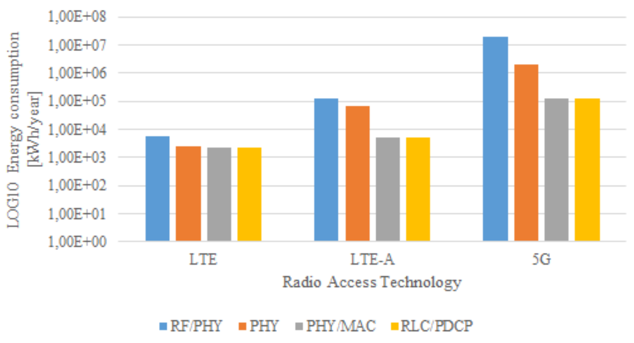


Figure 6. Energy consumption by RAT for one RU-CU connection using MWR FH.

D_{MAX} is the maximum distance between the CU and the RU due to FH latency constraints. RAT_n is the maximal transmission distance for one single antenna per Radio Access Technology (RAT). The RATs describe whether 3rd Generation (3G), LTE, LTE-A etc. are present in the current area, as each RAT requires its own RU. Equation (1) summarizes the amount of RUs for all RATs present, because different RATs have different cell sizes.

IV. Energy Consumption in the Physical Layer

This section introduces different options for physical layer FH transport and corresponding energy consumption models.

A. *PTP Fronthaul*

The traditional C-RAN FH network establishes a single PtP fiber transmission link between the CU and RU. The only energy consuming elements in this setup are the Small Form-factor Pluggable (SFP) modules and the media converter. The SFP modules are transceivers that connect the CU and RU to the fiber in each end of the connection, the media converter converts the signal to Ethernet before transmission.

1) *PtP energy consumption*

This work uses SFP+ modules for reference consuming 1.5 W each with a capacity of 10 Gbps [13]. For media converter a 10 Gbps capable media converter consuming 7.2 W is used for reference [14].

The amount of SFPs in PtP is dependent of the SFP capacity. For those FSs having a bitrate higher than the SFP capacity, the data stream will be split out on more channels, hence more SFPs. The number of SFPs per RU, SFP_{RU}, is calculated in (2), and is always at least one:

$$SFP_{RU} = \left\lceil \frac{B_{FH}}{Cap_{SFP}} \right\rceil \text{ , } SFP_{RU} \in \mathbb{N} \qquad (2)$$

In (2) B_{FH} is the FH bitrate, which can be found in Fig. 2, and cap_{SFP} is the capacity per SFP. The total number of SFPs can be calculated as, SFP_{PtP}.

$$SFP_{PtP} = SFP_{RU} \cdot N_{RU} \cdot 2 \text{ , } SFP_{PtP} \in \mathbb{N}_0 \qquad (3)$$

The power consumed by PtP can then be calculated as P_{PtP}, including the power per SFP, P_{SFP} and the power per media converter, P_{MC}, which is the same amount as the SFPs:

$$P_{PtP} = SFP_{PtP} \cdot P_{SFP} + SFP_{PtP} \cdot P_{MC} \text{ , } P_{PtP} \in \mathbb{R} \qquad (4)$$

2) *PtP Results*

One PtP connection consumes 17.4 W for FS RLC/PDCP. This corresponds to 152.4 kWh per year. Fig. 5. Illustrates the energy consumption for one PtP fiber CU-RU connection. The figure illustrates how the energy consumption is the same for all FSs in LTE. But when considering LTE-A and 5G bitrates, the energy consumption starts to scale. The decrease in energy consumption between RF/PHY split and PHY/MAC split is 99.3 % for 5G and 93.8 % for LTE-A.

B. *MWR Fronthaul*

The MWR FH establishes a wireless PtP connection between the CU and RU. In a MWR link, the RU is connected via a cable to the baseband processing part, which is then connected to one or more Transceivers (TRX), depending on required FH capacity. Each of the TRXs are then connected to an antenna. The antenna transmits radio waves to another antenna located close to the CU. This antenna is connected to a TRX and a baseband processing part, connected to the CU. As each TRX requires another TRX at the receiving side, these will in the following be referred to as TRX pairs. The energy consuming elements in a MWR FH consists of the baseband processing part and the TRXs in each end of the link.

1) *MWR energy consumption*

To provide information about MWR energy consumption, energy measurements were performed using an Ericsson Mini link MWR setup [15]. The system power was measured using a Rohde and Schwarz power Analyzer HMC8015. The configuration considered in this example is 28 MHz bandwidth and 1024-QAM modulation [15]. The baseband processing and one TRX in each end of the link consumes in total 250 W, and additional TRX pairs add 20 W on top of that [15]. These numbers assume that the TRXs are transmitting at the highest power. The capacity of each TRX

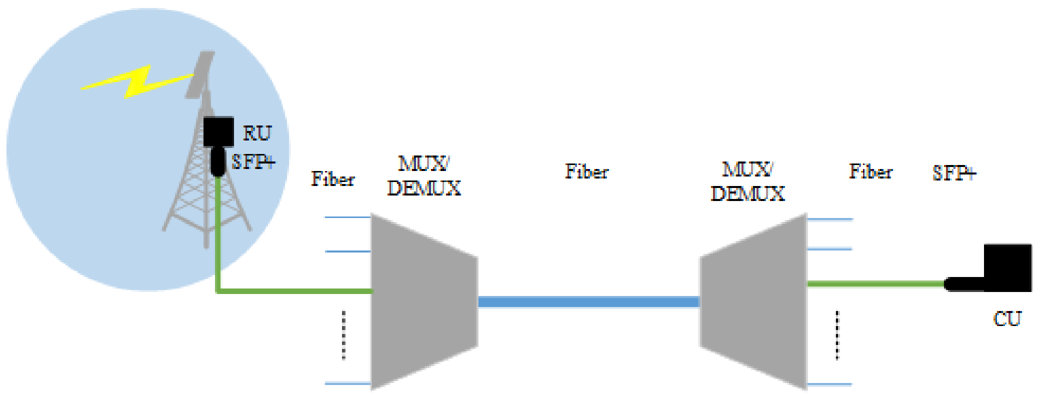


Figure 7. The CWDM system, multiplexing several fibers into one, and demultiplexing them again.

pair is 225 Mbps [15]. Studies show [15] that the power consumption of MWR links does not increase significantly with the user load. But another factor to consider is the transmission power, which when increasing also increases the energy consumption. The transmission power depends on the distance between the antennas, obstacles and weather conditions.

In MWR each CU-RU connection is a wireless PtP connection. The capacity necessary for transmission of one link is described by the number of TRX pairs. Calculating the total number of TRX pairs T, is done using the FH bitrate B_{FH} and the max capacity per TRX pair TRX_{cap}:

$$T = \left\lceil \frac{B_{FH}}{TRX_{cap}} \right\rceil , T \in \mathbb{N}_0 \qquad (5)$$

The power consumption for the whole MWR FH, P_{MWR} is then calculated using the power consumption of the baseband with one TRX pair included P_{BB} and the power consumption of additional TRX pairs, P_{TRX}:

$$P_{MWR} = P_{BB} + (T-1) \cdot P_{TRX} , P_{MWR} \in \mathbb{R} \qquad (6)$$

2) MWR Results

Fig. 6 illustrates the energy consumption for one RU-CU connection using the bitrates provided in Fig. 2. Note Fig. 6 uses a logarithmic scale on the Y-axis. The figure shows the large differences in energy consumption between the different RATs. In 5G the difference between RF/PHY split and RLC/PDCP split is a 99.3 % decrease, where in LTE the same decrease is only 59.7 %.

C. CWDM Fronthaul

CWDM is a multiplexing technique utilizing several optical wavelengths in the same fiber [16]. This is a very efficient way to transport data over great distances, as the only components requiring power along the transmission path are optical amplifiers [16]. CWDM transports up to 18 channels and for up to 60 km [16]. A CWDM network consists of a multiplexer that can combine data on several wavelengths, SFP modules, a fiber connection, one or more amplifiers along the path and a demultiplexer splitting the

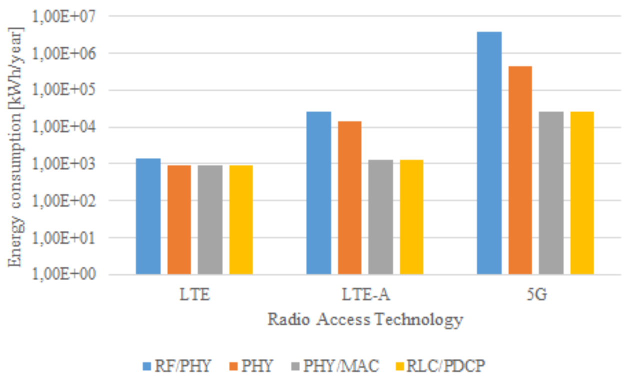


Figure 8. Energy consumption by RAT for one RU-CU connection using PON FH.

wavelengths out on the fibers connected to the individual RUs. The setup is illustrated in Fig. 7. Energy consuming elements in CWDM are the SFP modules, the media converters and the amplifiers along the transmission. The fiber medium does not consume any power itself and neither do the (de)multiplexers.

1) CWDM energy consumption

CWDM (de)multiplexers made with the newest technologies are passive and does not consume any energy themselves [17]. The transceivers considered for this CWDM system are CWDM SFP+ modules [13]. The CWDM 10 Gbps SFP+ modules consume 1.5 W each [13]. The media converter is 10 Gbps capable and consuming 7.2 W [14].

The energy consumption model for CWDM is equal to the one for PtP. This is due to the fact that the (de)multiplexers do not consume any power themselves, the energy consumption corresponds to the same as PtP.

2) CWDM Results

The CWDM power consumption corresponds to the power consumption in PtP.

D. PON Fronthaul

A PON network consists of one Optical Line Terminal (OLT) connected by fiber to one or multiple Optical Network Terminals (ONT). For this work the PON version Gigabit-

PON (GPON) is considered [18], this standard supports a 1.2 Gbps datarate on the FH link [18]. The process of transporting data from the OLT to the ONT is performed on a "transmit when there is something to send" basis, because the OLT is aware of all ONTs. The process of transporting the user data from the ONT to the OLT is more complex in PON, and a process referred to as Dynamic Bandwidth Allocation (DBA) is used for uplink scheduling. The DBA process assigns bandwidth to the individual ONTs when they request it [18]. The issue when considering PON as a FH transport medium arises in the uplink scheduling delay. This problem can be solved by creating a request message in the CU and this way send it on an earlier state to the OLT [19].

1) PON energy consumption

The products used for PON reference in this work is a Nokia ISAM FX-8 switch acting as the OLT using a 16 port line card, which means that this one device handles 16 OLTs at the time. Each of these OLTs connects up to 32 ONTs. The switch's power consumption is not included in this physical layer section. The GPON linecard consumes 89 W and optics are SFP B+ modules consuming 0.6 W per ONT.

The ONT considered for this work is an ICOTERA GPON FTTH ONT i5800-series with a maximum power consumption of 14.4 W [20].

The PON FH energy consumption model is first calculating how many ONTs are necessary, per ONT-OLT connection in order to transmit the required FH bitrate, ONT_{pair}:

$$ONT_{pair} = \left\lceil \frac{BR_{FS}}{BR_{PON}} \right\rceil, ONT_{pair} \in \mathbb{N} \quad (11)$$

Where BR_{PON} is the PON maximum bitrate and BR_{FS} is the maximum FH bitrate of the current FS. Then the number of ONTs connected to each OLT is calculated, ONT_{OLT}:

$$ONT_{OLT} = \left\lfloor \frac{BR_{PON}}{BR_{FS}} \right\rfloor, ONT_{OLT} \in \mathbb{N} \quad (12)$$

Then the number of OLTs in PON are calculated, PON_{OLT}:

$$PON_{OLT} = \left\lceil \frac{N_{RU}}{ONT_{OLT} \cdot ONT_{pair}} \right\rceil, \; PON_{OLT} \in \mathbb{N}_0 \quad (13)$$

Here N_{RU} is the number of RUs in the area. Then it should be calculated how many linecards N_{LC} are necessary, where OLT_{LC} is the number of OLT ports on the linecard.

$$N_{LC} = \left\lceil \frac{PON_{OLT}}{OLT_{LC}} \right\rceil, \; N_{LC} \in \mathbb{N}_0 \quad (14)$$

Then it is calculated how many SFP modules are necessary:

$$SFP = (ONT_{OLT} + ONT_{pair}) + PON_{OLT}, \; SFP \in \mathbb{N}_0 \quad (15)$$

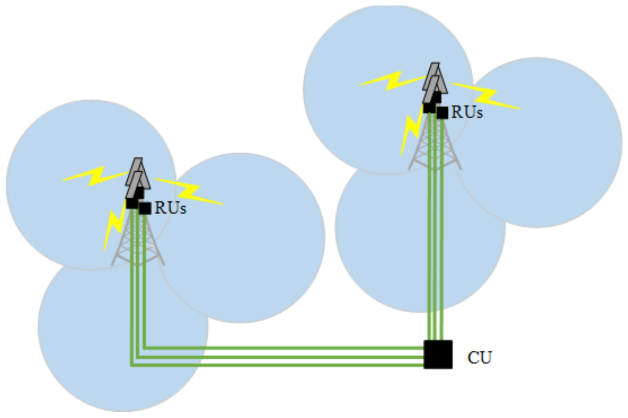


Figure 9. Illustration of several RUs connected to the same CU.

From this the total power consumption, P_{PON}, can be calculated:

$$P_{PON} = P_{LC} \cdot N_{LC} + P_{SFP} \cdot SFP + P_{ONT} \cdot (ONT_{OLT} + ONT_{pair}), \quad P_{PON} \in \mathbb{R} \quad (16)$$

In (16) P_{LC} is the power consumed by the linecard, P_{SFP} is the power consumed by each SFP module and P_{ONT} is the power consumed by each ONT.

2) PON Results

Fig. 8 illustrates the energy consumption by PON for one CU-RU connection considering different RATs and different FSs. Especially in the 5G scenario, the energy consumption is scaling much, here the decrease between split RF/PHY and PHY/MAC is 99.3 %. In LTE the decrease between the same FSs is 37.1 %.

E. Fronthaul networks comparison

This section considers a scenario where multiple RUs are connected to one CU. The situation is illustrated in Fig. 9 showing how multiple sites of 3-sectorized antennas are connected to one CU. In this use case, a FH network limited to 20 km by latency [7] using 3-sectorized antennas covering 13 km^2 per 3-sector, would need a total of 290 antennas/ RUs to cover the entire area calculated using (1). This example

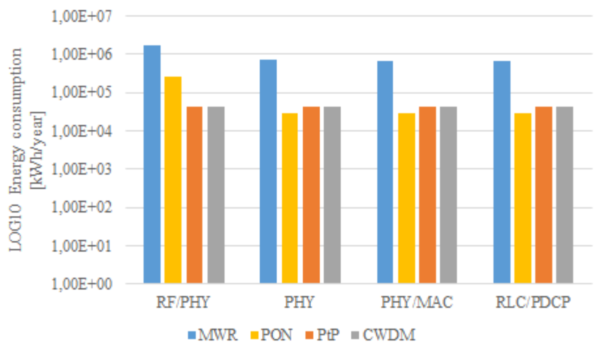


Figure 10. LTE FH energy consumption for different FH technologies considering an area with 290 RUs.

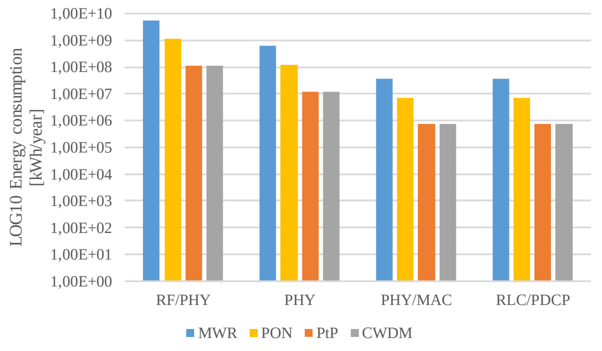


Figure 11. 5G FH energy consumption for different FH technologies.

only considers LTE as RAT. The results are summarized in Fig. 10 for LTE and Fig. 11 for 5G.

For both LTE and 5G, the largest energy consumption is consumed by the MWR connections, and the smallest energy consumption is consumed by PtP/CWDM. If changing from MWR FH to PtP/CWDM FH the energy consumption will be halved, regardless of the FS used. It is though interesting to observe that for the smallest FH bitrates in LTE, then PON is the least energy consuming solution.

V. ETHERNET FRONTHAUL

Ethernet is a data link layer transmission technology, which can act as overlay on top of the physical layer technologies already presented in this paper. Where the physical layer is just acting as pipes, Ethernet switches are forwarding Ethernet frames in the intended direction. As a FH network, Ethernet benefits in being flexible and already widely used in other network segments. Table II summarizes three options for FH transmission. The option of transmitting FH data using CPRI; this option is most beneficial for FSs located between the RF and the resource element mapper function, i.e., FSs having a constant bitrate on the FH link [4]. The same FSs can be transported over Ethernet using a gateway to encapsulate CPRI into Ethernet frames; this is referred to as CPRI over Ethernet. Another solution is FH transmission over Ethernet. This solution is preferred for FSs with a variable bitrate on the FH link, i.e., those having the resource mapper function included in the RU. Table II represents the current status of the network – the RUs connected using CPRI, and the Ethernet solutions as an option for the future 5G FH network.

The amount of RUs found in (1) can be used to find the estimated number of switches covering the current area. Hence, each RU is connected to one ingoing port in an Ethernet switch. Equation (17) expresses the lowest number of switches N_{sw} to cover an area:

$$N_{sw} = \left\lceil \frac{N_{RU}}{N_{port}} \right\rceil, N_{sw} \in \mathbb{N}_0 \qquad (17)$$

In (17), N_{port} is the number of ingoing ports in each switch. Fig. 12 illustrates the composition of an Ethernet switch. An Ethernet switch consists of different components. Ethernet frames are received in input modules, which type depends on whether the network is optical or electrical. Then the Ethernet frames are sent into the switch via receiving ports. When entering the switch, the Frame Check Sequence (FCS) is checked and the frame is stored in a FIFO queue. Then the address field in the frame is read, and matched in an address lookup process to the right outgoing port. Afterwards the frame is again stored in a FIFO queue, before it is sent to the outgoing ports and transmitted via output modules. All of these processes consumes energy depending on the FH link bitrate found in Fig. 2.

A. *Energy consumption*

The calculations in this paper uses a Cisco Catalyst 9200 switch for reference. This switch has a standby power of 35 W [21]. The switch has a power consumption of 42,27 W in case of full port traffic and 100% load [21]. The difference between standby and full load is thereby 7.27 W. Dividing this number into four switch processes, those mentioned in the previous sub section (FCS, MAC, FIFO, FIFO), a rough assumption is that each process consumes 1.8 W. The switch is assumed to use 24 ports running 1 Gbps speed and transmitting/receiving via SFP+ modules consuming 1.5 W each [22].

TABLE II. COMPARISON OF CPRI, ETHERNET AND CPRI OVER ETHERNET FRONTHAUL.

	Selected FH options		
	CPRI	***CPRI over Ethernet***	***Ethernet***
What is transmitted?	Raw IQ samples.	IQ samples encapsulated in Ethernet frames.	Ethernet frames.
Quality of Service	Dedicated user channel.	Shared transmission. Ethernet control management necessary.	Shared transmission. Ethernet control management necessary.
Pros	Simple RU. Capacity, timing and synchronization are guaranteed.	CPRI RUs can be reused. Existing Ethernet network can be used.	Variable/lower bitrate on FH link. Existing Ethernet network can be used.
Cons	Constant high bitrate on FH link increasing by number of antennas.	High Bitrate. Delay can occur. Requires a gateway from CPRI to CPRI over Ethernet.	Delay can occur. Requires new RUs with higher complexity.

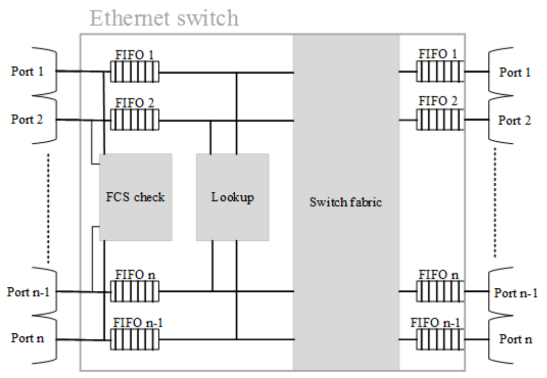


Figure 12. The construction of an Ethernet switch

B. *An Ethernet fronthaul energy consumption model*

In an Ethernet FH network, each switch consumes energy related to the amount of incoming traffic. This is expressed in (18), where P_{FH} is the total power in W consumed when transmitting data over the FH network between the RU and CU. P_{SW} is the total power consumed by one switch, and that is multiplied by the number of switches N_{SW} [1].

$$P_{FH} = P_{SW} \cdot N_{SW} \quad , P_{FH} \in \mathbb{R} \tag{18}$$

Equation (19) determines the power consumed in one switch. $P_{standby}$ is the power always consumed in the switch to keep it running. P_{pk} is the power consumed by the switch when forwarding one packet. P_{bit} is the power consumed by the switch when forwarding one bit. These numbers are multiplied by the number of forwarded bits, N_{bit}, and the number of forwarded packets, N_{pk}. These numbers are dependent of the FH bitrate, which can be found in Fig. 2 [1].

$$P_{sw} = P_{standby} + P_{pk} \cdot N_{pk} + P_{bit} \cdot N_{bit} \quad , P_{sw} \in \mathbb{R} \tag{19}$$

Determining the power consumed by the switch when forwarding one packet, requires the power consumed by the process only used once per packet, namely the MAC address lookup (P_{MAC}). This function's power consumption is divided by the maximal number of packets forwarded per second [1].

$$P_{pk} = \frac{P_{MAC}}{N_{pk}(\max)} , \; P_{pk} \in \mathbb{R} \tag{20}$$

N_{pk}(max), the maximal number of packets forwarded per second, is calculated by dividing the switch's maximum line bitrate by the minimum packet size.

Determining the power consumed by the switch when forwarding one bit, requires the power consumed by the processes where each bit is handled, namely the reception (P_{RX}), the FCS check (P_{FCS}), two FIFOs (P_{FIFO}) and the transmission (P_{TX}). These functions power consumption is divided by the maximal number of bits forwarded per second [1].

$$P_{bit} = \frac{P_{RX}+P_{FCS}+P_{FIFO} \cdot 2+P_{TX}}{N_{bit}(\max)} \; , P_{bit} \in \mathbb{R} \tag{21}$$

N_{bit}(max), the maximal number of bits forwarded per second, is the switch's maximum line bitrate. The given model is used for further investigation of the energy consumption in an Ethernet FH network.

C. *Results*

Based on the bitrate numbers provided in Fig. 2 and the Ethernet FH energy consumption model, are the following results obtained, illustrated in Figs. 13-17.

Fig. 13 illustrates the input parameters from the model in Section IV. The numbers are based on FS RF/PHY using 5G RAT for one switch. The energy consumption is illustrated on a logarithmic scale as a function of different packet sizes. The figure illustrates how different sizes of packets do not affect the total energy consumed by all bits (Pbit*Nbit) and neither the standby power (Pstandby) this is as expected as none of these parameters are affected by increasing packet sizes. However, the energy consumed for all packets (Ppk*Npk) is much affected by different packet sizes. The decrease in energy consumption between transmitting only the smallest possible Ethernet packets, and only the largest possible Ethernet packets is 95.78%.

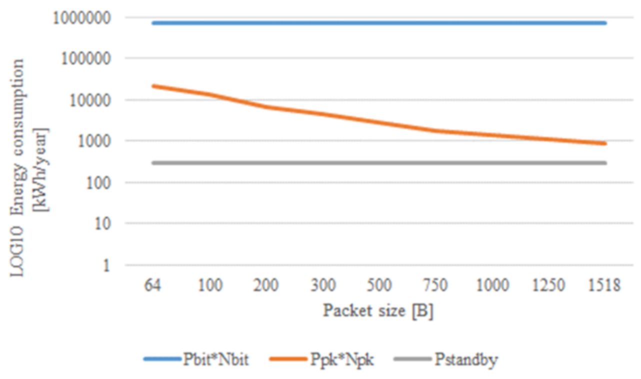


Figure 13. Energy consumption by packet size for the different elements in (3).

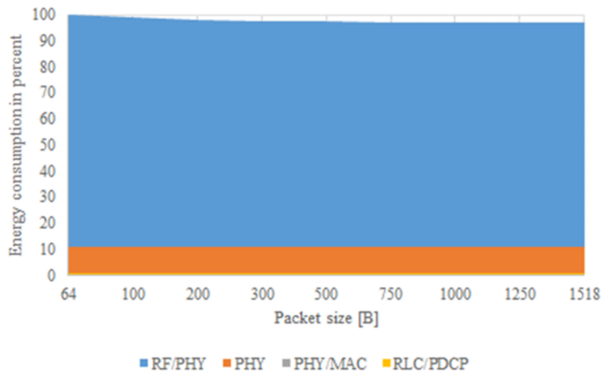


Figure 14. Percentage of energy consumption by increasing packet sizes using 5G RAT.

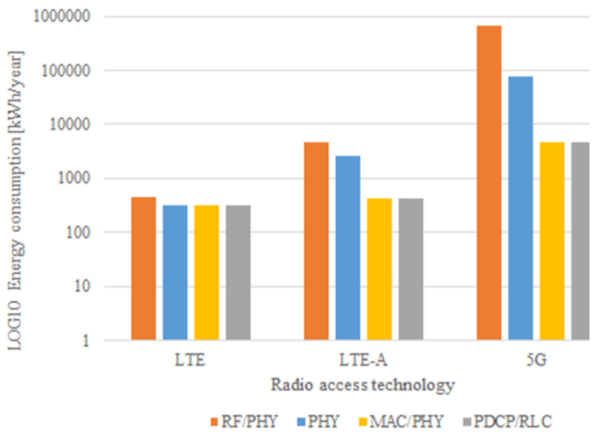


Figure 15. Ethernet energy consumption by radio access technology.

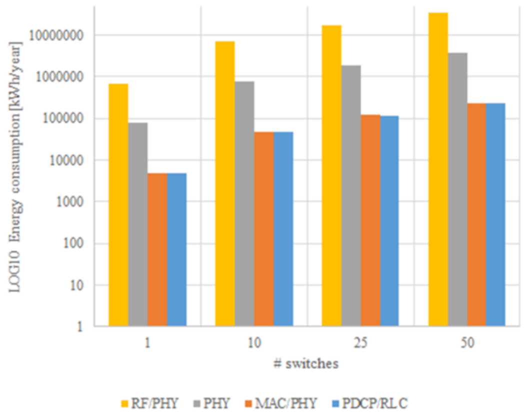


Figure 16. Energy consumption by increasing number of switches in 5G.

Fig. 14 illustrates the percentage of total switch energy consumption as a function of the packet sizes. The figure illustrates different FSs using 5G RAT. It is clear that the RF/PHY split consumes the largest percentage of energy. The figure shows how large effect the packet size has, thus the energy consumption percentage decreases slightly when the packets are larger. It is not possible to see the FSs PHY/MAC and RLC/PCP in the figure as they consume much less energy. However, in those splits the decrease in energy consumption between transmitting only the smallest possible Ethernet packet, and only the largest possible Ethernet packet is 2.66% in both cases whereas for the RF/PHY split the difference is 2.84%.

Fig. 15 illustrates the energy consumption in the FH network when using different FSs and different RATs. Note that it is illustrated on a logarithmic scale. This calculation assumes that the packet size is 1518 B. The figure shows the energy consumption in the FH network using LTE, LTE-A and 5G RATs. The figure states huge differences in power consumption for the different FSs using LTE-A and 5G. In 5G, the energy saving by using split PDCP/RLC compared to split RF/PHY is 99.32% per switch, compared to LTE where the energy saving is only 27.66% between the two splits. Or in other words if assuming one household consumes 3500 kWh per year, then the FH energy consumption in 5G using split RF/PHY covers 199 households per switch, where split PDCP/RLC covers less than 1.5 households per switch. In Fig. 15, the power consumption for LTE does not differ much when comparing the different FSs, meaning that significant energy consumption reductions or increases will not be present using this RAT.

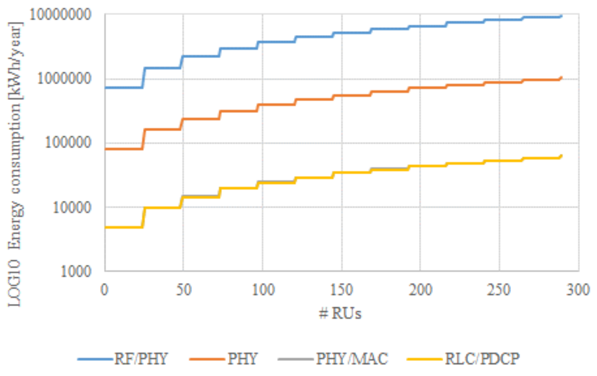


Figure 17. Energy consumption for 5G and different FS options compared to amount of RUs.

Fig. 16 shows the yearly FH energy consumption in kWh using 5G RAT. This calculation assumes that the packet size is 1518 B. The energy consumption is illustrated by the increasing number of switches. As the figure shows, then the Energy consumption increases by number of switches in the network. The figure illustrates how much energy is required to run a FH network with many switches.

Fig. 17 illustrates on a logarithmic scale, how the FH energy consumption increases when more RUs are added to the network. In the figure, each switch is assumed to have 24 ingoing ports, and the indent behavior of the graph shows the capacity of each switch.

VI. Fair inclusion of layer PHY tecnologies

The section aims to make the most fair comparison of the different PHY layer technologies using Ethernet transport. In general the comparison is difficult as PtP/CWDM results consider a 10 Gbps capacity system where the PON technology is only represented by a 1 Gbps capacity. But as numbers for energy consumption are difficult to find, it has been chosen to use confirmed energy consumption values and not estimated ones. For PtP/CWDM the Ethernet switch energy consumption is just added to the layer PHY energy consumption. For MWR, the energy consumption of the Ethernet switch is not included, as the MWR system transports Ethernet frames, and in that regard, energy consumption of Ethernet is already included in the numbers provided. For PON the SFPs have already been included in the calculations for the PON energy consumption, therefore the SFP energy consumption of the Ethernet switch P_{RX} and P_{TX} has been excluded from the calculations. Further, for

PON, the number of Ethernet switches is equal to the number of linecards, N_{LC}.

Figs. 18 and 19 illustrate the huge impact of the bitrate in these measurements, because the tendencies in the two figures are very different. The only difference between these two figures is the used FS and hereby the FH bitrate. According to Fig. 2, then the bitrate is more than 10 times larger for RF/PHY than for RLC/PDCP considering LTE. This difference is significant for PON. Because in the RLC/PDCP scenario the FH bitrate is smaller than maximum PON capacity, resulting in $ONT_{OLT} > 1$, meaning the capacity per OLT is shared among several ONTs. This results in PON having the lowest energy consumption in Fig. 19, whereas in Fig 18 $ONT_{pair} \leq 1$ and PON has the largest energy consumption. In this regard it would have been very interesting to compare with a 10 Gbps PON technology, but that must be a matter for future work.

VII. Discussion

The energy consumption is an important matter considering all areas of the ICT sector. The mobile networks are no exception and within mobile networks the FH network must never be a bottleneck for the expensive RAN capacity, but neither should it consume more energy than necessary. In that regard, the FH network must be carefully aligned. This work considers different layers of FH network transport. Different technologies in the physical layer and Ethernet in the data link layer. The total FH energy consumption will be the sum of the energy consumption in the physical layer and the energy consumption by the Ethernet overlay.

This work has looked into the energy consumption of different types of FH networks. In this regard, PtP fiber and CWDM definitely consume less energy compared to the other options. CWDM and PtP obtains very low results for energy consumption in this work, while it still have lots of capacity for the FH transmission. CWDM can be a solution to share resources in PtP scenarios because multiple links can be added to the same fiber. In reality CWDM will most likely be used to multiplex data from all RUs at the same base station onto the same fiber.

When comparing the different FH network types in Section VI, MWR consumes the largest amount of energy. At the same time, the MWR results are the only ones based on real life measurements, and not data sheet numbers. The MWR results depend on the bitrate and it is very clear to see how much the FS and the RAT influence the energy consumption. MWR might not be the most optimal solution energy-wise, but in some situations it is not possible to deploy fiber. Then MWR is an easy deployment solution.

PON as a FH network obtains in general higher energy consumption compared to the other wired solutions presented in this work, but not for lower bitrates as stated in Section VI, where the PON network is able to share the connections among several RUs. It should be noted that the energy consumption numbers provided in this work are only for GPON with a capacity of 1.2 Gbps, other PON solutions with higher bitrates, for example 10 Gbps or 40 Gbps, will most likely, according to Section VI, obtain much better results for energy consumption, because more resources can be shared among different users or more RUs. Unfortunately, it has only been possible to obtain energy consumption number for GPON for the examples in this paper. But, when having the power consumption for other PON solutions, the models in this work can still be used.

Considering the multiple RUs use case in Section IV.E, where an area covered by 290 RUs were used as an example. Here, changing from MWR FH to PtP/CWDM FH the energy consumption will be halved, regardless of the FS used. This is a significant difference from the one RU-CU pair examples presented in section IV.A,B,C,D. But in these examples lower decreases in energy consumption were provided using LTE RAT. More specifically, PtP and MWR had the same energy consumption for all FSs, MWR had a decrease between split RF/PHY and PHY/MAC of 60% whereas is PON the decrease between the same FSs were 37%.

Ethernet results in this work show how the choice of a FS, the number of RUs and the number of ingoing ports per Ethernet switch has huge impact on the energy consumption in an Ethernet FH network. The energy consumption does not differ much between the different FSs when considering LTE,

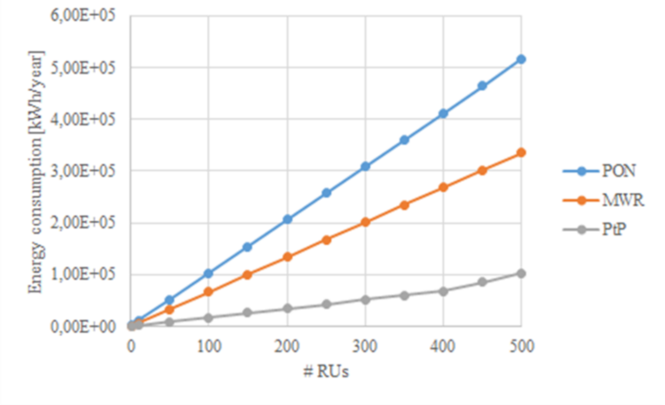


Figure 18. Ethernet energy consumption by increasing number of RU's .For FS RF/PHY and LTE RAT.

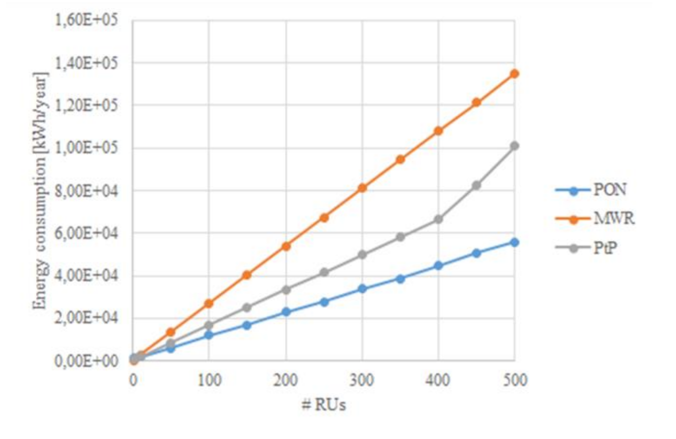


Figure 19. Ethernet energy consumption by increasing number of RU's. For FS PHY and LTE RAT.

but when entering the era of 5G, the FH networks will suffer from large energy consumption. To lower the energy consumption in the FH network, the choice of a FS becomes very important, together with high capacity Ethernet switches, and packet sizes. Slight decreases are obtained by transmitting larger sized packets even in splits PHY/MAC and PCP/RLC. In the Ethernet FH energy consumption model, a fixed packet size is used, which is very optimistic. In reality packets will be of different sizes, and the smaller packets, the more packets are necessary to transmit the same amount of data. At the same time, every packet carries a header, so more packets means more headers. Hence, using smaller packets, more bits have to be transmitted. In relation to that, it might not always be possible to fill up an entire Ethernet packet. Some functions in the protocol stack are time critical, e.g., the HARQ process [23]. In a time critical transmission, the packet might need to be sent before it is filled, leading to smaller packets and more overhead transmission.

This work provides an overview of the energy consumption of different network types. Products from different vendors or with higher capabilities may vary from these numbers provided. This work only considers the energy consumption, not the sustainability in using the resources already deployed. Or in other cases, one solution might be the only one possible, deployment-wise. Network operators can use the models provided in this work to estimate when it will be cheaper – energy-wise, to change already established equipment. Also for the case of green field deployment, the models provided can be used to evaluate the energy consumption of different solutions.

The results representing 5G and the extremely high bitrates and energy consumption related to that is only an extrapolation, but is found useful as a guideline for what can be expected.

VIII. Conclusion

This work investigated energy consumption in different types of FH networks for current and future mobile networks. The FH network connects the RU at the antenna site and the CU located in a datacenter. Different models for the FH energy consumption were presented, relating to different FH types. The outcome of this work shows the extremely high differences in energy consumption for different network types. Further, it is also important to choose the right FS, as significant reductions in energy consumption can be obtained. Many assumptions have been made due to lack of data but the paper gives an overview of the energy consumption in different network types and for the choice of different FSs. This makes the comparisons provided less accurate. But the models provided can be used for all vendors equipment and own numbers can easily be inserted. Results show that MWR consumes a lot of energy while CWDM consumes much less. If changing from MWR FH to PtP/CWDM FH the energy consumption will be halved, regardless of the FS used. For the Ethernet overlay alone, the difference in energy consumption between the, energy consumption wise, best and worst case scenario is 99.3 % per switch.

Acknowledgement

The authors would like to thank the Danish network provider Norlys for providing information about products and energy consumption in their PON and PtP networks.

The authors would also like to thank master's student Keld Fernstrøm for providing information about the energy consumption of MWR connections.

References

[1] L. P. Larsen, M. Berger, and H. Christiansen, "Energy-Aware Design Considerations for Ethernet-Based 5G Mobile Fronthaul Networks," *Proc. Fourth Int. Conf. Green Commun. Comput. Technol.*, 2019.

[2] N. Jones, "How to stop data centres from gobbling up the world's electricity," *https://www.nature.com/articles/d41586-018-06610-y . Accessed: June 2019*.

[3] N. Piovesan, A. Fernandez Gambin, M. Miozzo, M. Rossi, and P. Dini, "Energy sustainable paradigms and methods for future mobile networks: A survey," *Comput. Commun.*, vol. 119, pp. 101–117, Apr. 2018.

[4] L. M. P. Larsen, A. Checko, and H. L. Christiansen, "A Survey of the Functional Splits Proposed for 5G Mobile Crosshaul Networks," *IEEE Commun. Surv. Tutorials*, vol. 21, no. 1, pp. 146–172, 2019.

[5] NTT DOCOMO, "R3-162102: CU-DU split: Refinement for Annex A." 3GPP, 2016.

[6] NGMN, "5G RAN CU-DU Network Architecture, Transport Options and Dimensioning v 1.0," 2019.

[7] A. Checko *et al.*, "Cloud RAN for Mobile Networks - A Technology Overview," *IEEE Commun. Surv. Tutorials*, 2015.

[8] J. Liu, S. Xu, S. Zhou, and Z. Niu, "Redesigning fronthaul for next-generation networks: beyond baseband samples and point-to-point links," *IEEE Wirel. Commun.*, vol. 22, no. 5, pp. 90–97, Oct. 2015.

[9] M. Fathy, B. Mokhtar, M. A. Abdou, and M. R. M. Rizk, "Extended study towards performance improvement of Cloud-RAN," in *2017 13th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2017.

[10] Y. Alfadhli, M. Xu, S. Liu, F. Lu, P.-C. Peng, and G.-K. Chang, "Real-Time Demonstration of Adaptive Functional Split in 5G Flexible Mobile Fronthaul Networks," in *Optical Fiber Communication Conference*, 2018.

[11] Z. Tan, C. Yang, J. Song, Y. Liu, and Z. Wang, "Energy consumption analysis of C-RAN architecture based on 10G EPON front-haul with daily user behaviour," in *2015 14th International Conference on Optical Communications and Networks (ICOCN)*, 2015.

[12] K. Kondepu, "Performance Evaluation of SDN-Controlled Green Mobile Fronthaul Using a Federation of

Experimental Network," *Photonic Netw. Commun.*, vol. 37, no. 3, pp. 399–408, 2019.

[13] FS, "10G CWDM SFP+ 1470nm~1610nm 40km DOM Transceiver Data Sheet." [Online]. Available: https://img-en.fs.com/file/datasheet/cwdm-sfp-plus-1470nm-1610nm-40km-transceiver-datasheet.pdf . [Accessed: 20-Jan-2020].

[14] Perle, "S-10G Media Converter." [Online]. Available: https://www.perle.com/products/media-converters/10-gigabit-standalone-media-converters.shtml. [Accessed: 20-Jan-2020].

[15] Keld Fernstrøm, "Investigating the Relationship between Quality of Service and Energy Efficiency in Wireless Backhaul Radio Links," Master's thesis. Technical University of Denmark, 2020.

[16] E. Metsälä, *Mobile Backhaul*. Chichester, UK: John Wiley & Sons, Ltd, 2012.

[17] FS, "18ch Dual Fiber 1270-1610nm CWDM Mux Demux + Monitor Port Data Sheet." [Online]. Available: https://img-en.fs.com/file/datasheet/18ch-cwdm-mux-demux-1270-1610-with-mon.pdf . [Accessed: 20-Jan-2020].

[18] ITU-T, "ITU-T G.984.1: Gigabit-capable passive optical networks (GPON)," 2008.

[19] T. Tashiro *et al.*, "A Novel DBA Scheme for TDM-PON based Mobile Fronthaul," in *Optical Fiber Communication Conference*, 2014.

[20] Icotera, "GPON FTTH ONT i5800-series." [Online]. Available: https://icotera.com/media/1400/gpon-ont-i5800.pdf. [Accessed: 20-Jan-2020].

[21] Cisco, "Cisco Catalyst 9200 Series Switches Data Sheet." [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/switches/catalyst-9200-series-switches/nb-06-cat9200-ser-data-sheet-cte-en.html#Warranty. [Accessed: 01-May-2019].

[22] Cisco, "Cisco 10GBASE SFP+ Modules Data Sheet." [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/interfaces-modules/transceiver-modules/data_sheet_c78-455693.html. [Accessed: 01-May-2019].

[23] 3GPP, "TS 36.321 V15.6.0 (2019-06) Evolved Universal Terrestrial Radio Access (E-UTRA); Medium Access Control (MAC) protocol specification," 2019.

An Approach for Modelling Wireless Sensor Networks: Focusing on the Design Concept and Energy Awareness

Michel Bakni, Guillaume Terrasson, Octavian Curea, Alvaro Llaria, Jessye Dos Santos
Univ. Bordeaux, ESTIA Institute of Technology
F-64210 Bidart, France
e-mail: {m.bakni, g.terrasson, o.curea, a.llaria, j.dossantos}@estia.fr

Abstract— **In the design stage, Wireless Sensor Network developers generally need simulation tools to save time and money. These simulators require accurate models to precisely describe the behaviors of network nodes. Nevertheless, although model complexity has grown from layered-stack to cross-level, the energy aspects are not yet well implemented. In this paper, we suggest an energy-aware cross-level model for Wireless Sensor Network. Our modelling approach allows parameters that belong to different levels to interact and affect each other. This approach is used to predict the nodes energy consumption and to estimate the lifetime of the system. First, the results obtained from the implementation of our approach will be compared with those collected from a well-known simulator, Network Simulator version 2 using a set of basic scenarios. Then, the utility of our approach in the Wireless Sensor Network design process is highlighted using detailed scenarios that cover different types of interactions.**

Keywords-Energy-aware design; Cross-level; Energy modelling; Wireless Sensor Networks.

I. INTRODUCTION

A Wireless Sensor Network (WSN) is a set of battery-powered nodes that include sensors coupled with processing units and wireless transceivers. Since energy stored in the batteries is limited, both the autonomy of the node and the system's lifetime are impacted. Thus, energy-aware design is an important research topic and different solutions have been proposed in recent years to address the design concept from an energy point of view [1][2]. In this context, and to achieve the high-energy efficiency required by WSN to be implemented in domains like agriculture, industry, and healthcare [3], developers of WSN applications and researchers working in this field need to make the right decisions in the early stages of the design process. Consequently, and in order to deal with this challenge, simulators and emulators are widely used.

The scientific literature presents a large variety of single-level simulators, especially node-level or system-level [4]. This distribution can be traced back to the models on which the simulators were built. From what we studied, there is no WSN model designed to reveal the impacts of a given parameter on energy consumption from a cross-level perspective, and not only from a specific-level point of view [5]. Therefore, in [6], we briefly addressed the need of the cross-level design in WSN modelling. Later, we suggested a cross-level energy-aware model for WSN where interactions among model parameters were also highlighted, but only for the radiofrequency (RF) unit onboard the node [1]. In this paper, the application of the model is expanded to include other node components, precisely the processing unit and a temperature sensor, together with more detailed examples of cross-level interaction.

The structure of this paper is as follows. First, the evolution of WSN modelling is reviewed focusing on design concept and energy-awareness. Then, we propose a cross-level energy-aware model for WSN. After that, the result obtained from the implemented model is compared with results from a well-known WSN simulator: NS2. Additionally, a set of scenarios will be presented to show cross-level interactions among model parameters. This includes distance between the nodes and fragmentation threshold. Finally, we conclude with the perspective and future works.

II. WSN MODELLING AND SIMULATION

In order to better understand the main challenges related to WSN modelling and simulation in a power-aware design context, a short study on the evolution of WSN models is first provided. Throughout, the development of modelling techniques is traced. Then, WSN simulators, which were built on the previous model, are to be classified based on two characteristics: design concept and energy-awareness.

A. WSNs Models: from Layered-stack to Cross-level Design

Classical data network models, such as the Open Systems Interconnection (OSI) model or the Internet model (TCP/IP), suggest a layer-based design approach where a set of layers are stacked together. In this kind of model, the layers are defined based on their functionality, such as physical or logical connections. Each layer can only interact with the two adjacent layers in the stack using software interfaces [7]. However, in WSNs, parameters residing at different layers need to interact with each other [3]. Thus, services are to be provided across stack layers rather than a specific layer. Thereby, the traditional stack-layered model cannot answer the modelling challenges of this kind of networks [8].

For many years, several works have handled this problem by developing different WSN-oriented solutions. In [7], the authors suggest a modelling approach based on cross-layer design, where adaptivity and optimization across multiple layers of the stack is supported. A comparable cross-layered model is introduced in [9]. Furthermore, in [10], the cross-layer point of view is applied to network security, and additional considerations are included in this regard. In [11] a cross-layer concept is suggested through an infrastructure that supports a cross-layer design approach. According to the authors, the cross-layer interaction means that some data from a layer may be used as a parameter by a protocol that resides in another layer or it may affect another layer's operating process.

The suggestion of the "tier" or "level" concept is another approach that is used to answer the modelling challenges in WSN [12]. In the previous proposal, a level stands for a group of parameters that belong to different functions and features of the entire system, and not only limited to the network model. Alongside the previous methods, the solution outlined in [13] divides the model's layers into two levels. The first level is mainly focused on non-physical parameters related to both software and application. On the other hand, the second level is concerned with hardware where the protocols are implemented, especially linking and routing, along with RF unit and sensor parameters.

This resulted in the development of a new approach for modelling WSNs: the multi-level approach. It expands the standard layered-stack model to cover not only node hardware and software, but also wireless media and parameters from the surrounding environment, such as temperature. Examples of this kind of multi-level models can be found in [14] and [15].

After that, several evolutions were suggested to develop a cross-level model starting from the multi-level approach. As an example, in [16], a description is provided for using the multiple-level model in the design and development of WSN from a cross-level perspective. However, as illustrated in [17], many proposed cross-level approaches are not used in an effective manner. Consequently, although energy modelling is a crucial issue in WSN design, this aspect is also not well implemented in the modelling stage.

B. *WSN Simulator classification: design concept and energy awareness*

In general, WSN models focus on one level of abstraction. Therefore, WSNs simulators built on these models are typically specific level [2]. Consequently, they trace parameters related to that particular level [3]. In this context, a parameter is a numeric value that characterizes one property of a given level of abstraction, such as power consumption or payload length.

For example, Network Simulator version 2, well-known as NS2 [17], is aimed at simulating network protocols. Thus, it has poor support for hardware simulation. Quite the opposite, TOSSIM [18] can precisely emulate hardware, but it provides a very abstract perspective for high-level network or routing protocols. Moreover, there are several multi-level simulators. In this category, the simulator can take into account parameters belonging to different levels simultaneously. By way of an example, Jsim [19] is a multi-level simulator dedicated to WSNs. It is multi-level because it can simulate both environment and network parameters at the same time. Finally, few simulators are classified as cross-level because they have the ability to show cross-level interaction among parameters belonging to different levels of abstraction. COOJA [20] is an example of this kind of simulator.

None of the simulators mentioned above is energy oriented. These simulators could trace energy parameters, but they were not really built purpose for that. Based on that, a level-based energy classification for WSN simulators is proposed in [4] and illustrated in Fig. 1. In this figure, simulators are first classified according to their energy-awareness: energy-oriented and non-energy-oriented. This classification is extended to include the simulator design concepts that are listed above.

Looking at energy-oriented simulators, PowerTOSSIM [21] is an extension of TOSSIM that is dedicated to the emulation of energy in hardware. There is also IDEA1 [22]. It is a multi-level energy-oriented simulator that handles three abstract levels: the wireless medium, the node and the environment.

The previous review shows that several non-energy-oriented simulators, based on multi or cross-level models have been suggested. However, as demonstrated, regarding the energy aspect, the cross-level approach is not implemented. Therefore, with existing simulators, it is difficult to analyze the influence of parameters belonging to different levels of abstraction on both WSN lifetime and total energy consumption in a given node.

III. Cross-level Approach for Energy aware Modelling

Considering the previously mentioned limitations of existing simulators, we propose a cross-level approach for modelling WSN in the energy-aware context. First, the main definitions on which the model is built are introduced. After that, a global overview of the suggested concept is provided. Based on that, the relationship between parameters within each node component is explained to demonstrate the cross-level interactions.

A. *General definitions*

As previously mentioned, we start by defining the principles on which the proposed concept is based.

Definitions relating to the design concept are first introduced:

- Parameter: a configurable value that represents a specific property of the level it belongs to.
- Level: an abstract design concept. It stands for a set of parameters that describes the same part of the

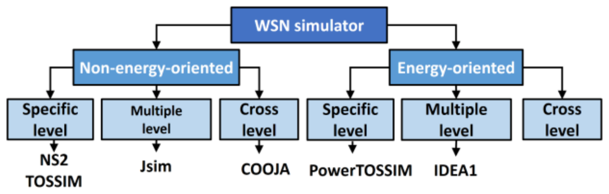


Figure 1. Energy-aware level-based classification for WSN simulators

system that is being modeled.

- Interaction: a relationship between two parameters that mutually affect each other. If the parameters belong to the same level, it is to be called an intra-level interaction. Otherwise, if the two parameters reside at different levels, the interaction is to be called cross-level.

Secondly, definitions related to the energy-aware design process are provided:

- Phase: a time period corresponds to one circuit activity, each phase is combined with a current consumption level.
- Pattern: a set of subsequent phases for several circuits. Each pattern corresponds to a particular node and it describes its behavior in terms of current consumption and time.
- Pattern frequency (F_P): the number of pattern occurrences in one second.

B. *Global Concept*

Regarding the definitions above, the global overview of our cross-level is illustrated in Fig. 2. Based on that, a WSN is divided into four separate levels of abstraction. Within each level, a set of parameters describes the level's properties. Interactions between the parameters can be cross-level or intra-level. A short description of each level is provided below.

- The Use case Level (UL) is the highest level in the concept hierarchy. It is concerned with the WSN application requirements. Parameters such as pattern frequency (Fp) or node activity sequences reside at this level.
- The System Level (SL) reflects the topological aspect of the WSN. It focuses on how algorithms and high-level protocols influence the performance of WSNs. Parameters related to the network topology, like the distance between nodes, or protocols specifications, such as fragmentation threshold belong to this level.
- The Node Level (NL) is concerned with the interactions among node software, such as the operating system, and onboard hardware, for example, the RF unit. In other words, this level focuses on the interactions between the node components. The parameters relating to the structure of node patterns, such as the order of the phases reside at this level.
- The Circuit Level (CL) is the lowest level in the model's hierarchy. It is used particularly to characterize node hardware. The modelling of the electronic circuits takes place at this level, and this is where hardware-specific parameters, such as the power levels or the supply voltage, reside. The circuit level includes an RF unit, a processing unit, a set of sensors, and a battery.

Therefore, as the four levels are stacked together, the higher the level is, the more general the parameters are. For example, parameters belonging to the UL are related to the scenario description, and they theoretically fit into any WSN application. On the other hand, CL parameters are very specific, they describe particular electronic circuits.

C. *Cross-level Interactions for Node Circuits*

In this section, the interactions, both intra-level and cross-level, between parameters that concern each circuit are explained.

As the proposed concept is energy-aware, in the following section, we focus on the energy aspects of WSN. Energy consumption in WSN is circuit-based, i.e., each circuit consumes energy independently. However, consumption is governed by the activities of the circuits which, in turn, depend on the interactions between parameters.

In the description below, next to each parameter's name, the abbreviated name of the level to which the parameter belongs will be added.

1) *RF unit*

The proposed concept is first applied to describe RF activities in a WSN. Fig. 3 provides an overview of all the interactions that takes in the different stages of RF activities.

In the first stage, the total number of bits to send is calculated. This includes the payload (UL) created by sensors or other applications that generates data on the node, as well as high-level protocol headers (SL) and the link layer protocol header (SL). Note that interactions at this stage are both intra-level and cross-level.

After that, the total amount of data is confronted against the fragmentation threshold (SL) identified by the wireless link protocol. If fragmentation is needed, the process will take place in this stage, and will result in two or more data frames. Next, the preamble (SL) is added to each data frame. Then, the length of each frame is calculated in term of seconds, thanks to the bitrate(s) (SL) supported by both the link protocol and the selected RF chipset.

The next step is to build the energy consumption pattern

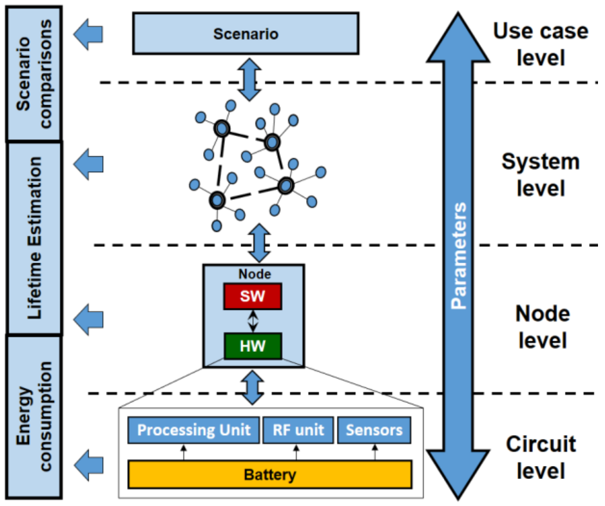


Figure 2. The Proposed Cross-level concept for Wireless Sensor Network

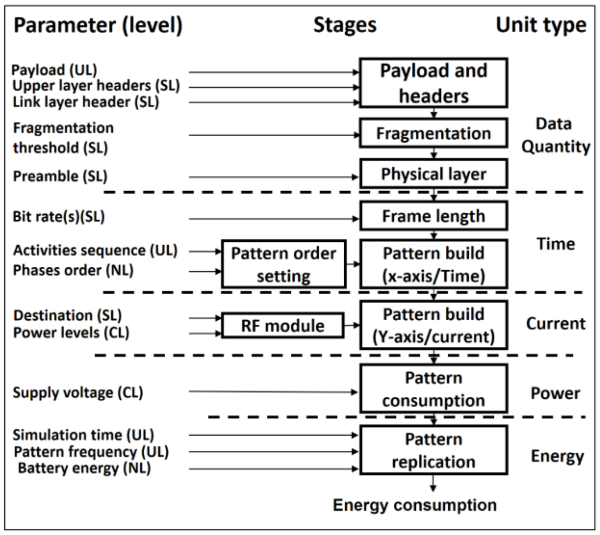


Figure 3. Cross-level design for parameters interaction in the RF unit

for the node's RF activities. Activity sequences (UL) and phase order (NL) are to be considered. The activity sequence helps to specify the actions taking place, such as sending or receiving frames. The link protocol defines the phases within each activity, as well as their order. For example, in the sending activity, the order of phases as follows: accessing the channel (ph_{acc}), exchanging data frames (ph_{exch}) and then waiting for acknowledgment (ph_{ack}).

After that, the sequence of activities is to be matched with the power levels (CL) provided by the RF unit datasheet. This includes considering the distance (SL) that has a direct impact on the power level of the sending activity. Then, considering the supply voltage of the RF circuit (CL), the energy consumed by the RF activities is calculated.

In the final stage, the total simulation time (UL), the frequency of the pattern F_p (UL), as well as the initial amount of the energy stored in the battery (NL) are taken into consideration in order to estimate the system's lifetime.

2) *Processing Unit*

All the activities relating to the processing unit are node-based. Interactions only occur with other onboard components. However, the processing unit controls the RF unit's activities in terms of wake-up and sleep times. Additionally, it manages the activities of the sensors in the same way. Thus, synchronizing the node's activities is the main task of the processing unit.

The time sequence for the processing unit pattern starts in the sleep phase, in which, all node components are in sleep state. Then, the unit wakes up and enters the active phase where all other component activities take place. Finally, the processing unit return to the sleep state. Fig. 4 shows an abstract pattern of the processing unit, where the three phases: sleep, wake-up, and active take place, respectively. As illustrated, the amount of time spent in each phase depends on cross-level and intra-level interactions between the parameters.

After that, the processing unit's pattern will be created. To achieve that, the corresponding power level (CL) for each phase is derived from the circuit datasheet.

As a result, and considering the supply voltage (CL), the energy consumption of the processing unit in one pattern is calculated. Finally, the simulation time (UL), frequency pattern (UL) and battery energy (NL) are added as well, and the effects of the processing unit's activities on the node's lifetime can be estimated. Fig. 5 displayed in the sequential stages is used to calculate energy consumption in the processing unit.

3) *Sensing Unit*

Sensors interact with the physical environment and with other nodes' components only, which means there will be no inter-node activities. Fig. 6 provides an overview of all the interactions that take place in one sensing unit. Note that multiple sensing units can exist onboard the WSN node.

When sensing a physical phenomenon (SL), the sensor measures a physical quantity and converts the measured value into an electric signal. The conversion time (CL) is specified in the sensor datasheet provided by the manufacturer. This is the time when most energy is consumed by the sensor. We consider that the result of this stage is a set of bits captured from the sensor's environment.

The sensor's activity sequence (UL) is then to be

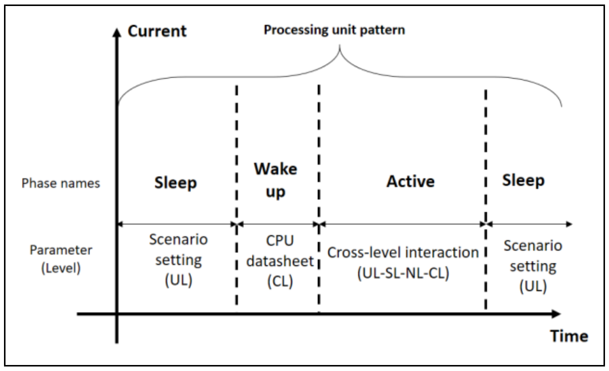


Figure 4. Pattern construction and cross-level design for parameters in the processing unit

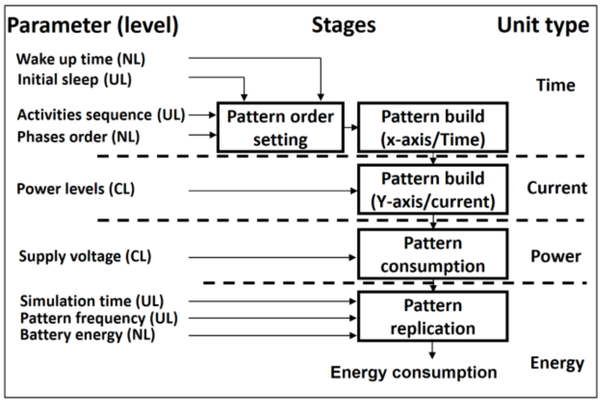


Figure 5. Cross-level design for parameters interaction in the processing unit

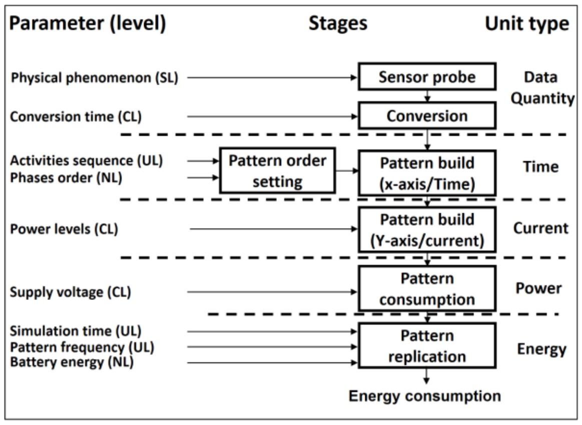


Figure 6. Cross-level design for parameters interaction in the consumption in the sensing unit

included. This includes the number occurrences in which the measurements take place in one pattern, as well as intervals between measurements. When the sensor is not sensing, it is in sleep mode. The result of this stage is the time sequence of the sensor's phases.

Next, the power level (CL) derived from the sensor datasheet is included. For each phase, there will be a corresponding consumption level. After that, the battery's voltage will be added, and energy consumption for one pattern of sensor activities is calculated. Finally, simulation time, pattern frequency and the energy level are to be included, thus, an estimation for the effects of the sensor's activities on the node's lifetime can be generated.

D. *Model design*

As illustrated previously, the proposed model is divided into four levels of abstraction. Going one step further in the modelling process, we introduce the block. It is an abstract modelling object that exists in the level's boundary. Each block includes a set of parameters. Blocks connect with each other using one of the following types of relationships:

1) Deriver-source connection: this connection links two blocks: the derivative and the source. The derivative block inherits all source attributes and behavior. Moreover, it has additional extensions that create variants of the source to serve the derivation purpose.
2) Child-parent connection: it couples two blocks: the child and the parent. The child block is part of the parent, which can have more than one child block of the same type.

The blocks can be classified based on the level in which they reside. Fig. 7 shows a block diagram for the proposed model, the levels are separated using dashed lines, and the highest level of abstraction is at the far right of the diagram. Blocks are represented using a rectangle. Additionally, block connections are shown, as well as the number of possible children or parents for each child-parent connection. Note that interactions link the model's parameters, while connections represent relationships between the blocks.

Hereafter, starting from the highest level of abstraction, there is a short description of the blocks used in the modelling process.

- UL blocks: UL includes one or more scenario blocks, each of which represents a full scenario. A given scenario includes scenario-level parameters such as the scenario time as well as blocks from the topology level like the network topology block. In the modelling hierarchy, the scenario block is the highest.
- SL blocks: this is the level where information relating to the whole system is parameterized. Examples of parameters are the number of nodes and the distance separating them. This level includes two topology-related blocks: network topology and wireless medium blocks. The topology block includes a set of node blocks, each of which represents one node in the network, and all those nodes are node-level blocks. The wireless medium characterizes the communication channel.
- NL blocks: these reside at the node-level and cover the parameters at that level, such as node position. There are three blocks at this level: node, type, and pattern blocks. The node block stands for the physical node in the modeled system. Each node is associated with a pattern and type blocks. The type serves a design requirement: heterogeneity. A type includes circuit blocks. Additionally, a type block can be shared among a set of nodes. The pattern block represents the node's periodic behavior. If two nodes share the same pattern, they will behave identically, in terms of activities (sending, sensing, etc.), but the consumption associated with each activity might be different based on other parameters, such as node positions. The type block also includes a battery, processing unit, RF unit, and sensors block. These are all circuit-level blocks.
- CL blocks: these describe the hardware components of the node. The battery block is a circuit-level block. It represents the physical battery and includes energy parameters, such as the battery's nominal values. The circuit is a circuit-level abstract block, i.e., other blocks can be derived from it, they are the processing unit, RF unit and the sensors, each of which describes one specific component. These blocks include parameters related to the physical circuits. The parameters can usually be obtained from the circuit's datasheets, like the power level. Finally, the RF unit block is associated with the protocol block. The protocol block describes the wireless link protocol used by the RF unit. It covers parameters related to the link protocol such as bit rates.

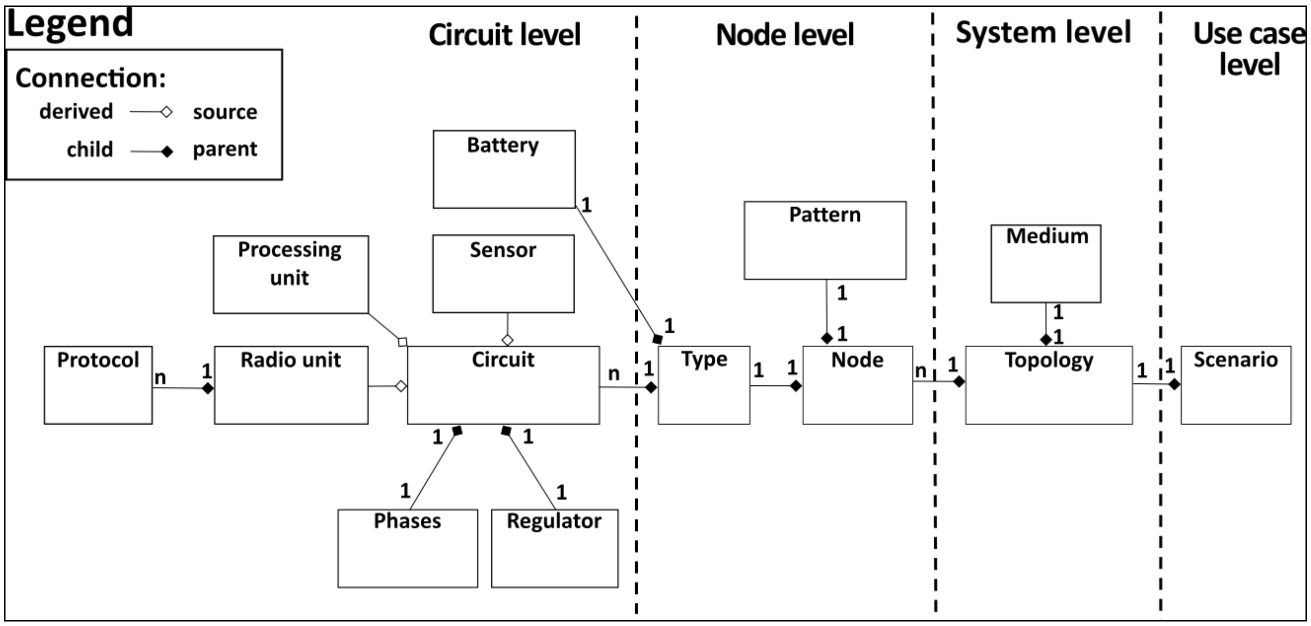


Figure 7. Block diagram for the proposed model

As mentioned above, the model supports network heterogeneity. This is a property describing the ability to simulate heterogeneous systems where different types of nodes can exist in the same scenario [23]. The connection between the type and the node blocks at the node level answers heterogeneity. Each node includes one type block, but they can vary in its components and protocols. Note that the pattern block is not a part of the type, which means that two identical nodes can have different energy consumption patterns, where each consumes energy based on its parameters. The distance to the destination parameter is an example of that. Different distances will create various patterns, thus, non-equal energy consumption among the nodes, although they might be identical in terms of components.

As we have seen in the WSN modelling, this is the first model to include circuit regulators. Although the benefits of this addition were not directly explained in this paper, the transferring from one unit to another when representing consumption, precisely from current to energy, was done thanks to the regulators. The regulator modelling allows to distinguish the voltage levels for each point on the node, thus, facilitating the consumption calculation and make it more accurate.

IV. Application of our model

A. General settings

Our model is implemented using Matlab. Then tested scenarios will be proposed. These are designed for two purposes. First, presenting interactions between parameters, both cross-level and intra-level. Second, comparing the results obtained from the proposed model with those of a well-known simulator, namely NS2.

All the scenarios take place in an open area where there are two wireless nodes named Node A and Node B. These settings for the scenarios where selected to show the interactions between the parameters. Thus, simple two-node scenarios were developed. Periodically, Node A sends a fixed-length payload to Node B through the wireless medium using its RF unit.

Each node applies a TCP/IP network model. The implementation of the protocols starts at the network layer where Internet Protocol version 4 (IPv4) [24] has been chosen. The Internet Control Message Protocol (ICMP) [25], which is an integral part of IPv4, is used to create echo messages when necessary. Consequently, when the IPv4 module receives a data packet, it is going to send the same data back to the original source. In all scenarios, the length of IPv4 and ICMP headers are 20 and 4 bytes, respectively.

Next, the link-layer parameters are to be set. Two different protocols are to be used, these are: IEEE 802.15.4 as [26] and IEEE 802.11a is used [27]. The energy specification for the IEEE 802.15.4 module is derived from the CC2420 transceiver (Texas Instruments) [28]. For IEEE802.11a the specifications will be derived from an implemented chipset named HDG204 (H&D wireless) [29]. Table I shows the settings for the two wireless link protocols. For each scenario, the considered time used to calculate energy consumption is 100 seconds, and it begins after initializing the nodes.

The previous settings are general. They are to be implemented in all the scenarios. However, in the following sections, additional settings are to be added will be explicitly mentioned.

B. Payload and pattern frequency

Pattern frequency and payload size are UL parameters. In this set of scenarios, we trace the cross-level effects of the two on the consumed energy, which is a CL parameter. First, we suggest a scenario using only the RF unit and we compare the obtained results with those from NS2. The reason behind this choice is the need to have comparable results, i.e., NS2 does not include models for other hardware components. Second, the scenario will be extended to include other hardware components, namely a processing unit and a sensor. This will highlight the contribution of the proposed modelling approach.

In order to illustrate the cross-level interaction, we go back to Fig. 7. When the payload size (UL) is modified, the length of the data frame to be sent (CL) by the RF unit will be changed. This, in turn, will impact the pattern structure (NL) and will have an influence on the total energy consumed by the node (CL), as well as the system's estimated lifetime (UL). The cross-level interaction for the frequency pattern can be traced in the same way. However, in this case, the change of F_P (UL) will directly affect the pattern structure (NL), without passing by the circuit level. Then, the chain interaction will follow the same way as the payload size interactions.

1) Scenarios with only RF unit

To effectively trace payload size and F_P, the following settings are to be added to the general settings. First, the two nodes will reside 10m apart. Second, both IEEE 802.11a IEEE 802.15.4 are to be used. For each protocol, three different values of F_p: 0.1, 1 and 2 Hz, and ten values for the payload length ranging between 10 to 100 bytes are used. Each scenario requires a combination of the three parameters previously mentioned. As a result, there are 30 scenarios to be run for each wireless link protocol. Table II summarizes the setting for the scenarios with only RF unit included.

These scenarios are configured both in NS2 and in the proposed model implemented in Matlab. The obtained results are the energy consumed by different activities of the RF unit. These activities are categorized into 4 phases:

- Access phase (ph_{acc}): RF unit tries to access the wireless channel.
- Exchange phase (ph_{exch}): RF unit sends or receives data frames.
- Acknowledge phase (ph_{ack}): RF unit sends or receives acknowledgment frames.
- Sleep phase (ph_{slp}): The RF unit is in sleep state.

In these phases, the cross-level interaction between parameters is taking place, i.e., the energy consumed in each phase is a result of interplay between parameters related to different levels. For example, in the exchange phase, the consumed energy depends on of the headers' lengths, payload length, and bit rate, which belongs to the following levels: UL, SL, and NL, respectively.

The results obtained from the implementation of the previously described scenarios in NS2 and in Matlab using our model can be found in Table III. The upper part shows the obtained results from IEEE 802.11a and the lower part for IEEE 802.15.4. Each part is further divided into two subparts, corresponding to scenarios with 10 and 100 bytes of the payload length, respectively.

For the two protocols, when comparing obtained results from NS2 and our proposed model, differences and similarities can be found as follows. The energy consumed in ph_{acc} or in ph_{ack} are identical and there is a slight difference in the energy consumed in ph_{slp}. In ph_{exch}, the difference is notable, but stable and this is due to different interpretations of the link protocol specifications. For example, in our model, the ICMP header is considered to be part of the data packet, contrary to NS2 where this header is added to the data packet later.

Fig. 8 shows the pattern obtained from the proposed model. The left side of the figure is dedicated to IEEE 802.11a and the right side to IEEE 802.15.4. Different values of F_p are displayed, namely 0.1, 1, and 2 Hz. For each of these values, a set of corresponding errors is provided. Each of these is also related to a simulation where the payload length is 10, 50, or 100 bytes.

Next, the relative error between the results obtained from NS2 and our model is to be calculated. For each scenario, the relative error is calculated as follows. The value obtained from out model is subtracted from the value obtained from NS2, and the result is to be divided by the latter. In all the simulation results, the relative errors obtained from IEEE 802.11a are greater than those of the corresponding scenarios of IEEE 802.15.4. This difference can be explained by unplanned and non-periodic radio activities that appear periodically in NS2 IEEE 802.11a simulations. These activities have a fixed duration regardless of F_p and the payload length. Each of these activities appears as a single pulse of transmission or reception causing an additional energy consumption of around 5 µJ and 2 µJ, respectively.

Finally, as illustrated in Fig. 9, the value of the relative error between NS2 and the proposed model did not exceed 3.5%. Based on that, we can consider the proposed model is validated with NS2. Although not shown in this scenario, our proposed WSNs model also allows adding energy consumption phases for other hardware on the node, such as the processing unit or sensors. As a result, an accurate pattern can be constructed representing precisely the real consumption of the node contrary to NS2 that has a poor support for hardware, as mentioned before.

TABLE I. WIRELESS LINK PROTOCOL SETTINGS

Parameter	IEEE 802.11a (HDG204)	IEEE 802.15.4 (CC2420)
Bitrate [bps]	12 M	250 K
Carrier Sense Mechanism	Pure CSMA/CA	CCA-ED
Transmitter power [mW]	725	52
Receiver Power [mW]	220	59
Sleep Power [mW]	0.2	0.06

TABLE II. GENERAL SETTINGS OF THE RF UNIT ONLY SCENARIOS

Parameter	Value
Number of the nodes	2
Node positions	(10,10), (10,20) [m]
Scenario duration T_{Sce}	100 [s]
Pattern Frequency F_p	0.1, 1, 2 [Hz]
Payload length	10, 20, …, 100 [Byte]
Link protocol	IEEE 802.11a, IEEE 802.15.4

2) Complete node Scenario

A typical WSN node includes other components in addition to the RF unit, such as processing unit and sensors. Together, they serve the node functionality. In order for the components to serve the node, they consume a considerable amount of energy that cannot be neglected. For example, energy profiling in [30] shows that the RF unit is responsible for 62% of the node consumption, where the sensors and the microcontroller share the remaining 38% of the energy consumed. Another study shows different energy profiles based on the scenario setting, the consumption of the RF unit varying between 7 to 65% of the total consumed energy, while the other components are responsible for the remaining consumption [31].

NS2 is not capable of tracking the energy consumed in sensors and the microcontroller. However, the proposed model is able to do that. This is simply achieved by extending the consumption pattern to include new phases for the other components.

In this scenario, the setting of the RF only scenario will be implemented considering the following extensions. We assume that the node includes a microcontroller and a sensor. The microcontroller is PIC18F4620 [32] and the sensor is TMP102 [33], a temperature sensor. The scenario is still the same, except for the following: The node wakes up after 0.5 seconds from the pattern start. Then, the sensor captures a value directly after the node is up it sends the data through the RF unit. Finally, it goes back to sleep again without waiting for the response. In addition to that, the size of the payload is only 2 bytes since the sensor creates a 12-bit sample.

All the circuits are supplied with 3.3 V. The electrical characteristic for the RF unit will not be changed. Regarding the processing unit and sensor, Table IV provides supply current in the active and sleep phases. Note that the wake-up duration of the processing unit lasts for 1 µs, during which we consider the supply current to be identical to the active phase.

In Fig. 10, a pie chart is shown for a per-component energy consumption percentage for the node A. The energy consumed by the processing unit is 60% of the total consumption and sensor consumes only 4% of the total energy. However, energy consumed by the RF unit is 36%. However, the selecting the hardware components, such as those using different processing units or the scenario settings, like activity sequences can significantly impact the energy profile.

In addition to that, the model is also capable of tracing the energy consumption down to the circuit level, i.e., the consumption of the phases for each circuit. Table V provides results obtained from the sending node, the energy consumed by the processing unit is divided into three phases: Sleep,

TABLE III. ENERGY CONSUMPTION OF THE PHASES IN DIFFERENT SCENARIOS (F_P = 1 Hz)

Simulation	**Consumed energy per phase [µJ]**				
	Sleep	**Exchange**	**Access**	**Ack.**	**Total**
IEEE 802.11a					
10 Bytes					
Matlab	51.03	50.08	23.63	199.959	324.69
NS2	51.03	55.76	23.63	199.950	330.37
100 Bytes					
Matlab	51.03	106.79	23.63	199.935	381.38
NS2	51.03	112.46	23.63	199.927	387.04
IEEE 802.15.4					
10 Bytes					
Matlab	18.89	145.73	39.07	59.78	263.47
NS2	18.88	152.84	39.07	59.78	270.47
100 Bytes					
Matlab	18.89	465.37	39.07	59.44	582.77
NS2	18.88	472.42	39.07	59.45	589.82

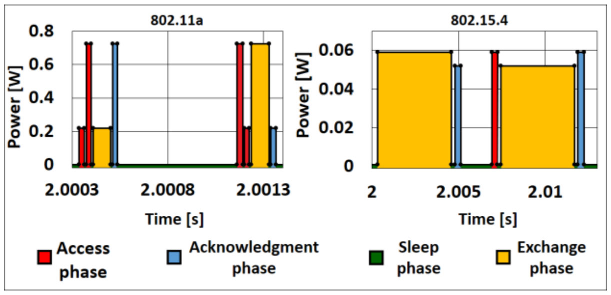


Figure 8. Energy consumption patterns for different wireless link protocols (Node B, Payload length = 100 Bytes, F_p = 1 Hz)

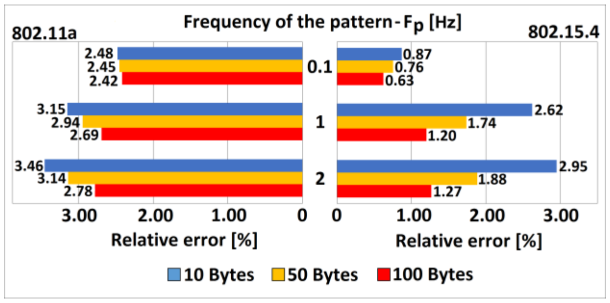


Figure 9. Relative error between the proposed model and NS2

TABLE IV. ELECTRICAL CHARACTERISTICS FOR THE PROCESSING UNIT AND THE SENSOR IN THE COMPLETE NODE SCENARIO

Phase	**TMP102**	**PIC16F4620**
Sleep [µW]	1.65	0.33
Active [µW]	280.5	4290

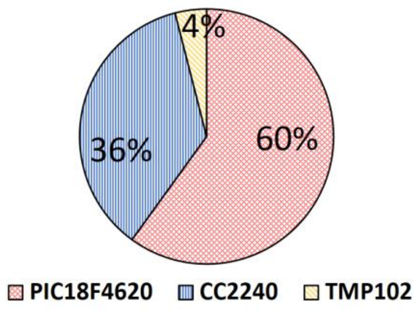


Figure 10. Per-component energy consumption profile for the complete node scenario

wake up and active, and consumption in the active phase has the higher value. For the sensor, the energy consumed is categorized into two phases: sleep and active. Following the same consumption trend, most consumption takes place during the active phase.

The proposed model answers the needs of WSN designers and researchers. Regarding the first group, the model provides a way to combine different circuits and protocols. This is useful in the application design stage where decisions relating to the construction of the node are to be made. On the other hand, the model can be used in the field of research as well, because it can show the cross-level interactions, and this is interesting for energy-related studies.

The suggested model is extensible, i.e., new blocks can be added to support new functionalities. For example, a wave propagation model can be added at the system level. Additionally, an energy harvester unit can extend the model at the circuit level. However, the model activities and the corresponding phases need to be added to the pattern at the node level as well.

C. *Distance and power levels*

Distance between nodes has an impact on energy consumption in the RF module. Manufacturers of the RF chipsets provide a range of transmission power levels. The further the destination is, the higher the transmission power level selected. The proposed model provides a way to trace this cross-level interaction.

The cross-level interactions are illustrated using arrows in Fig. 11. First, a distance between two nodes changes, it is a system-level parameter. This, in turn, interacts with the power level selection in the RF unit at the circuit level. Through intra-level interaction, the activity sequence in the RF unit is affected. This is illustrated using a blue arrow on the same figure. After that, the cross-level chain interaction reaches the node pattern at the node level. Finally, energy consumption in the battery at the circuit level, and thus, the system's lifetime at the scenario level are affected.

TABLE V. ENERGY CONSUMPTION FOR THE NODE COMPONENTS IN ONE PATTERN FOR THE COMPLETE NODE SCENARIO. (PAYLOAD = 2 BYTES)

Processing unit: PIC16F4620			
Phase	**Sleep**	**Wake-up**	**Active**
Energy [µJ]	0.359	4.81 x 10^{-3}	143.03
RF Module: CC2240			
Phase	**Sleep**	**Access**	**Exchange**	**Ack**
Energy [µJ]	1.31	17.8	40.75	24.48
Sensor: TMP102			
Phase	**Sleep**		**Active**
Energy [µJ]	1.8		7.69

In order to show the previous cross-level interactions, the general settings will be extended as follows. First, the payload length will be 20 bytes in all scenarios. Second, the distance between the two nodes will change to 25, 50, 100, 200, and 400 meters, respectively. Those values were intentionally selected to cover the whole power level range proposed in the RF unit's datasheet.

Fig. 12 shows the obtained values for energy consumption of the RF unit exchange phases from the distance and power level scenario. As the distance between nodes increases, the energy consumption in the exchange phase increases linearly. This can be explained from the chipset datasheet specification. As the distance rises above a particular threshold, the model automatically changes the transmission threshold to use the next value. This change has an impact on the energy consumed by the node. For example, when the distance between the two nodes changed from 25 to 400m, the corresponding consumed energy in the transmission phase changed from 60 to 120 µJ.

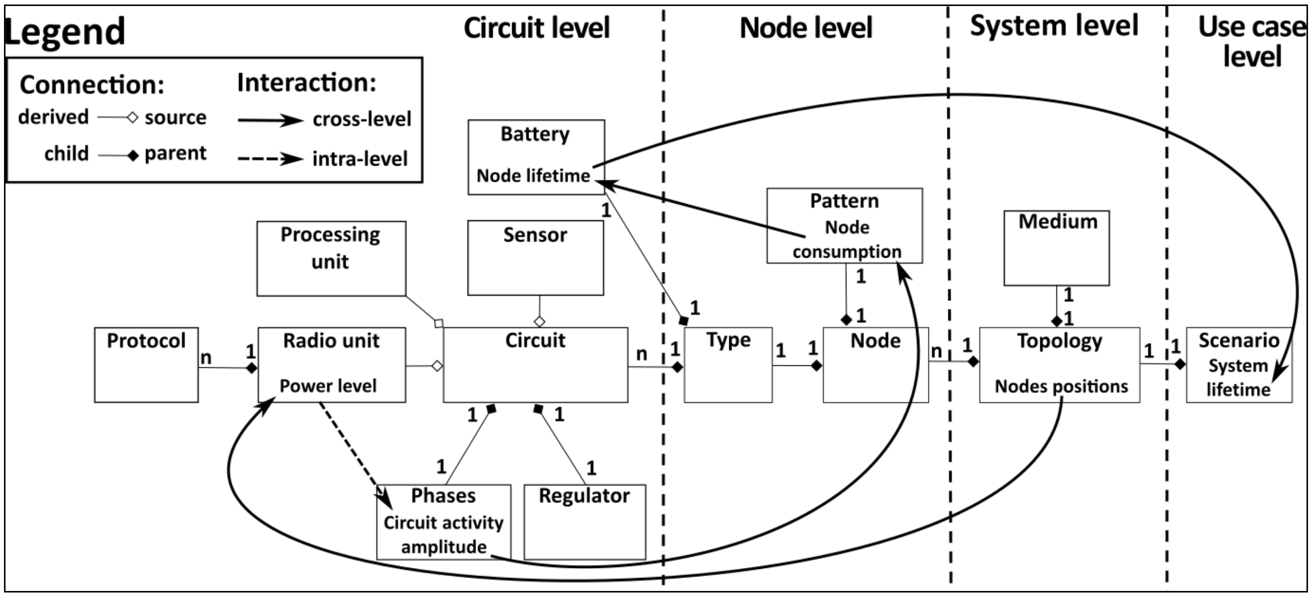


Figure 11. Block diagram for the proposed model

D. *Fragmentation threshold*

Fragmentation is a mechanism to divide Protocol data units; i.e., frames or packets, into two or more units that are shorter in length. Fragmentation is used when WSN applications contain a constraint on the size of data units. For example, IEEE 802.15.4 has a maximum fragmentation threshold set to 127 bytes, but any lower value down to one, can be set. Fragmentation threshold is a system-level parameter which impacts the energy consumption of the RF unit at the circuit level. The proposed model can be used to show the previous cross-level interaction.

In this case, the cross-level interactions take place as follows: first, the fragmentation threshold will be changed. It is a system-level parameter. This change will affect the protocol object, which is attached to the RF unit object, and both are circuit-level parameters. These changes will affect the pattern structure at the node level, which, in turn, affects the energy consumption in the node, which is a circuit-level parameter. In the end, the system's lifetime will be affected by this cross-level chain interaction.

In practice, the settings of the complete node scenario will be implemented. However, the following changes will take place. The scenario will include sending a payload of 100 bytes, and the fragmentation threshold is set to {40, 60, 80, 100} bytes, respectively. TMP102 is not used. It will always remain in the sleep state, thus, its consumption is constant in all the scenarios and it will be ignored. On the other hand, the consumption of the RF unit and the processing unit is changed as a response to the fragmentation setting, and these changes are to be traced. The node wakes up after 0.5 seconds from the pattern start and begins sending the data packets.

Fig. 13 presents the results obtained from the fragmentation scenario. The energy consumption by the RF unit phases is illustrated according to the fragmentation threshold. The consumption of access, exchange, and acknowledgment phases drop significantly after the first test. Energy consumption in the sleep phase changes slightly.

These results can be explained with the help of the number of frames sent in each scenario: they are 15, 4, 3, and 2, respectively. This sharp drop in the number of the sent frames is due to the effect of another system-level parameter, namely protocol headers. The used protocols need to add headers to the payload. It is 33 bytes in total, and that leaves only 7 bytes for the payload when the fragmentation threshold is set to 40 bytes. However, the payload size is up to 27 bytes when the threshold is 60 bytes. Thus, 100 bytes of payload need only 4 frames to be completely sent. In general, these results show that energy consumption in the node decreases as the fragmentation threshold increases.

V. CONCLUSION AND FUTURE WORK

In this paper, a cross-level energy-aware modelling approach for WSN is proposed. This approach was applied by mean of several scenarios to reflect how parameters from different levels can interact and affect energy consumption. The suggested model is built on the assumption that WSN node activities can be described in a pattern, which is periodically repeated. A pattern consists of sequential phases belonging to all the circuits composing the node.

Moreover, the obtained results show the model's ability to provide energy consumption at different levels of abstraction. This includes the total energy consumed by each node, the consumption of one specific circuit, as well as that of a particular phase. Furthermore, this capability will be extended in future work to include the estimation of the system's lifetime.

The characterization of energy consumption requires handling time on a very large scale. On the one hand, it is necessary to calculate energy consumption in the node pattern accurately using microsecond scale. On the other hand, the system's lifetime tends to be expressed in years. Our approach can answer those issues because the simulation time is a linear gain between the pattern energy and the total amount of consumed energy.

However, for the moment, our approach cannot integrate non-repetitive or unpredictable activities. For example, whenever the network topology changes during the scenario, routing protocols should be activated to find new routes and these activities will consume energy in an irregular way.

Currently, we are developing a WSN simulator based on the proposed approach. The simulator will be designed for both researchers and WSN application developers. In parallel, we are developing a test bench based on physical nodes. The objective is to compare the simulation results with measurement from real applications, when the same scenarios are implemented in both environments.

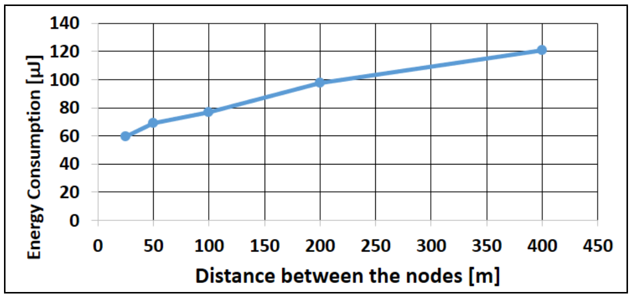


Figure 12. Energy consumption of the RF unit exchange phase in the distance and power level scenario

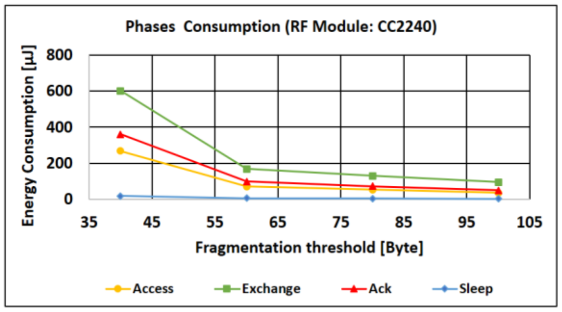


Figure 13. Per-phase energy consumption from the fragmentation scenario for the RF unit

ACKNOWLEDGMENT

The work presented in this paper has been financially supported by the Regional Council of New Aquitaine (Call for Projects 2016 funds), as part of the OUDINI research project.

REFERENCES

[1] M. Bakni, G. Terrasson, O. Curea, A. Llaria, and J. Dos Santos, "Energy-aware Cross-level Model for Wireless Sensor Networks," SENSORCOMM, 2019. pp. 46-51. ISBN: 9781612087443.

[2] R. Chéour, M.W. Jmal, O. Kanoun, and M. Abid, "Evaluation of simulator tools and power-aware scheduling model for wireless sensor networks," IET Computers & Digital Techniques, vol. 11, no. 5. 2017, pp. 173-182, doi: 10.1049/iet-cdt.2017.0003.

[3] J. Yick, M. Biswanath, and G. Dipak. "Wireless sensor network survey," Computer networks, vol. 52, no. 12 .2008. pp. 2292-2330. doi: 10.1016/j.comnet.2008.04.002.

[4] M. Bakni, L. Chacón, Y. Cardinale, G. Terrasson, and O. Curea, "WSN Simulators Evaluation: An Approach Focusing on Energy awareness," International journal of wireless & mobile networks (IJWMN), 2020, 11 (6), pp. 1-20. doi: 10.5121/ijwmn.2019.11601.

[5] J. Haase, J. M. Molina, and D. Dietrich, "Power-aware system design of wireless sensor networks: Power estimation and power profiling strategies," IEEE Transactions on Industrial Informatics, vol. 7, no. 4, 2011. pp. 601-613. doi: 10.1109/TII.2011.2166793.

[6] M. Bakni, G. Terrasson, O. Curea, A. Llaria, and J. Dos Santos, "A Cross-level model for power-aware Wireless Sensor Networks design," 13ème Colloque National du GDR SOC2, 2019.

[7] H. Zimmermann, "OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection," in IEEE Transactions on Communications. vol. 28, no. 4. 1980. pp. 425-432. doi: 10.1109/TCOM.1980.1094702.

[8] A. J. Goldsmith and S. B. Wicker, "Design challenges for energy-constrained ad hoc wireless networks," IEEE Wireless Communications, vol. 9, no. 4, 2002, pp. 8-27. doi: 10.1109/MWC.2002.1028874.

[9] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci, "Wireless sensor networks: a survey," Computer Networks, vol. 38, no. 4, 2002. pp. 393-422. doi: 10.1016/S1389-1286(01)00302-4.

[10] W. Charfi, M. Masmoudi, and F. Derbel, "A layered model for wireless sensor networks," 2009 6th International Multi-Conference on Systems, Signals and Devices, 2009, pp. 1-5. doi: 10.1109/SSD.2009.4956693.

[11] I. Korkmaz, O. Dagdeviren and M. E. Dalkilic, "A 2-hop coloring-based collision free infrastructure design for Wireless Sensor Networks," 2016 HONET-ICT, Nicosia. 2016, pp. 65-69, doi: 10.1109/HONET.2016.7753421.

[12] E. Egea-López, J. Vales-Alonso, A. S. Martínez-Sala, P. Pavón-Mariño, and J. García-Haro, "Simulation tools for wireless sensor networks," International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS05), Society for Modeling & Simulation International, 2005, pp. 1-8, ISBN: 9781622763504.

[13] M. Kuorilehto, M. Kohvakka, M. Hännikäinen, and T.D. Hämäläinen, "High abstraction level design and implementation framework for wireless sensor networks," International Workshop on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS 2005), Lecture Notes in Computer Science, 2005, pp. 384-393. doi: 10.1007/11512622_41.

[14] M. Korkalainen, M. Sallinen, N. Kärkkäinen, and P. Tukeva, "Survey of wireless sensor networks simulation tools for demanding applications," 2009 Fifth International Conference on Networking and Services, 2009, pp. 102-106, doi: 10.1109/ICNS.2009.75.

[15] W. Du, D. Navarro, F. Mieyeville, and F. Gaffiot, "Towards a taxonomy of simulation tools for wireless sensor networks," Proceedings of the 3rd International ICST Conference on Simulation Tools and Techniques, 2010. doi: 10.4108/ICST.SIMUTOOLS2010.8659.

[16] M. Imran, A.M. Said, and H. Hasbullah, "A survey of simulators, emulators and testbeds for wireless sensor networks," 2010 International Symposium on Information Technology, 2010, pp. 897-902. doi: 10.1109/ITSIM.2010.5561571.

[17] T. Issariyakul and E. Hossain, "Introduction to Network Simulator 2 (NS2)," in Introduction to Network Simulator NS2. Springer US, pp. 21-40, 2012, doi: 10.1007/978-0-387-71760-9_2.

[18] P. Levis, N. Lee, M. Welsh, and D. Culler, "TOSSIM: Accurate and scalable simulation of entire TinyOS applications," Proceedings of the 1st international conference on Embedded networked sensor systems, 2003, pp. 126-137, doi: 10.1145/958491.958506.

[19] A. Sobeih, J.C. Hou, L-C. Kung, N. Li, H. Zhang, W-P. Chen, H-Y. Tyan, and H. Lim, "J-Sim: A simulation and emulation environment for wireless sensor networks," IEEE Wireless Communications, vol. 13, no. 4. pp. 104-119. 2006, doi: 10.1109/MWC.2006.1678171.

[20] F. Osterlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt, "Cross-Level sensor network simulation with COOJA," Proceedings. 2006 31st IEEE Conference on Local Computer Networks, 2006, pp. 641-648. doi: 10.1109/LCN.2006.322172.

[21] E. Perla, A.Ó. Catháin, R.S. Carbajo, M. Huggard, and C. McGoldric, "PowerTOSSIM z: Realistic energy modelling for wireless sensor network environments," Proceedings of the 3nd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks, 2008, pp. 35-42. doi: 10.1145/1454630.1454636.

[22] W. Du, D. Navarro, F. Mieyeville, and I. O'Connor, "IDEA1: A validated system C-based simulator for wireless sensor networks," Proceedings of the 2011 IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, 2011, pp. 825-830. doi: 10.1109/MASS.2011.95.

[23] M. Bakni, L. Chacón, Y. Cardinale, G. Terrasson, and O. Curea. "Methodology to Evaluate WSN Simulators: Focusing on Energy Consumption Awareness". 6th International Conference on Computer Science, Engineering and Information Technology (CSEIT-2019), 2019. pp. 331-351, doi: 10.5121/csit.2019.91327.

[24] J. Postel, "RFC 791, Internet Protocol", 1981. doi: 10.17487/RFC0791.

[25] J. Postel, "RFC 792, Internet Control Message Protocol", 1981. doi: 10.17487/RFC0792.

[26] "IEEE Standard for Information technology -- Local and metropolitan area networks -- Specific requirements-- Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low Rate Wireless Personal Area Networks (WPANs)," in IEEE Std 802.15.4-2006 (Revision of IEEE Std 802.15.4-2003), pp. 1-320, 2006. doi:10.1109/IEEESTD.2006.232110.

[27] "IEEE Standard for Telecommunications and Information Exchange Between Systems - LAN/MAN Specific Requirements - Part 11: Wireless Medium Access Control (MAC) and physical layer (PHY) specifications: High Speed Physical Layer in the 5 GHz band," in IEEE Std 802.11a-1999, pp. 1-102, 1999. doi: 10.1109/IEEESTD.1999.90606.

[28] "2.4 GHz IEEE 802.15.4 / ZigBee-ready RF Transceiver", SWRS041c, Texas Instruments, 2019.

[29] "HDG204-WiFi 802.11b+g System in Package", 1451 - Rev. PA5 02/2011 , H&D Wireless AB, 2011.

[30] G. Terrasson, R. Briand, S. Basrour, V. Dupé, O. Arrijuria, "Energy Model for the Design of Ultra-Low Power Nodes for Wireless Sensor Networks", Procedia Chemistry, vol. 1, no. 1, 2009, pp. 1195-1198. doi: 10.1016/j.proche.2009.07.298.

[31] M. Navarro, Y. Li and Y. Liang, "Energy profile for environmental monitoring wireless sensor networks," 2014 IEEE Colombian Conference on Communications and Computing (COLCOM), 2014, pp. 1-6. doi: 10.1109/ColComCon.2014.6860416.

[32] "PIC18F2525/2620/4525/4620 Data Sheet 28/40/44-Pin Enhanced Flash Microcontrollers with 10-Bit A/D and nanoWatt Technology", DS39626E, Microchip Technology, 2008.

[33] "TMP102 Low-Power Digital Temperature Sensor With SMBus and Two-Wire Serial Interface in SOT563", SBOS397H, Texas Instruments, 2018.