International Journal on Advances in Internet Technology

2018 vol. 11 nr. 3&4

The *International Journal on Advances in Internet Technology* is published by IARIA.
ISSN: 1942-2652
journals site: http://www.iariajournals.org
contact: petre@iaria.org

Sponsored by IARIA
www.iaria.org

International Journal on Advances in Internet Technology
Volume 11, Number 3 & 4, 2018

Xi Chen, University of Washington, USA
IlKwon Cho, National Information Society Agency, South Korea
Andrzej Chydzinski, Silesian University of Technology, Poland
Noël Crespi, Telecom SudParis, France
Antonio Cuadra-Sanchez, Indra, Spain
Javier Cubo, University of Malaga, Spain
Sagarmay Deb, Central Queensland University, Australia
Javier Del Ser, Tecnalia Research & Innovation, Spain
Philipe Devienne, LIFL - Université Lille 1 - CNRS, France
Kamil Dimililer, Near East Universiy, Cyprus
Martin Dobler, Vorarlberg University of Applied Sciences, Austria
Jean-Michel Dricot, Université Libre de Bruxelles, Belgium
Matthias Ehmann, Universität Bayreuth, Germany
Tarek El-Bawab, Jackson State University, USA
Nashwa Mamdouh El-Bendary, Arab Academy for Science, Technology, and Maritime Transport, Egypt
Mohamed Dafir El Kettani, ENSIAS - Université Mohammed V-Souissi, Morocco
Armando Ferro, University of the Basque Country (UPV/EHU), Spain
Anders Fongen, Norwegian Defence Research Establishment, Norway
Giancarlo Fortino, University of Calabria, Italy
Kary Främling, Aalto University, Finland
Steffen Fries, Siemens AG, Corporate Technology - Munich, Germany
Ivan Ganchev, University of Limerick, Ireland / University of Plovdiv “Paisii Hilendarski”, Bulgaria
Shang Gao, Zhongnan University of Economics and Law, China
Kamini Garg, University of Applied Sciences Southern Switzerland, Lugano, Switzerland
Rosario Giuseppe Garroppo, Dipartimento Ingegneria dell'informazione - Università di Pisa, Italy
Thierry Gayraud, LAAS-CNRS / Université de Toulouse / Université Paul Sabatier, France
Christos K. Georgiadis, University of Macedonia, Greece
Katja Gilly, Universidad Miguel Hernandez, Spain
Mariusz Głąbowski, Poznan University of Technology, Poland
Feliz Gouveia, Universidade Fernando Pessoa - Porto, Portugal
Kannan Govindan, Crash Avoidance Metrics Partnership (CAMP), USA
Bill Grosky, University of Michigan-Dearborn, USA
Jason Gu, Singapore University of Technology and Design, Singapore
Christophe Guéret, Vrije Universiteit Amsterdam, Nederlands
Frederic Guidec, IRISA-UBS, Université de Bretagne-Sud, France
Bin Guo, Northwestern Polytechnical University, China
Gerhard Hancke, Royal Holloway / University of London, UK
Arthur Herzog, Technische Universität Darmstadt, Germany
Rattikorn Hewett, Whitacre College of Engineering, Texas Tech University, USA
Quang Hieu Vu, EBTIC, Khalifa University, Arab Emirates
Hiroaki Higaki, Tokyo Denki University, Japan
Dong Ho Cho, Korea Advanced Institute of Science and Technology (KAIST), Korea
Anna Hristoskova, Ghent University - IBBT, Belgium
Ching-Hsien (Robert) Hsu, Chung Hua University, Taiwan
Chi Hung, Tsinghua University, China
Edward Hung, Hong Kong Polytechnic University, Hong Kong
Raj Jain, Washington University in St. Louis , USA
Edward Jaser, Princess Sumaya University for Technology - Amman, Jordan
Terje Jensen, Telenor Group Industrial Development / Norwegian University of Science and Technology, Norway
Yasushi Kambayashi, Nippon Institute of Technology, Japan
Georgios Kambourakis, University of the Aegean, Greece
Atsushi Kanai, Hosei University, Japan
Henrik Karstoft , Aarhus University, Denmark

Dimitrios Katsaros, University of Thessaly, Greece
Ayad ali Keshlaf, Newcastle University, UK
Reinhard Klemm, Avaya Labs Research, USA
Samad Kolahi, Unitec Institute Of Technology, New Zealand
Dmitry Korzun, Petrozavodsk State University, Russia / Aalto University, Finland
Slawomir Kuklinski, Warsaw University of Technology, Poland
Andrew Kusiak, The University of Iowa, USA
Mikel Larrea, University of the Basque Country UPV/EHU, Spain
Frédéric Le Mouël, University of Lyon, INSA Lyon / INRIA, France
Juong-Sik Lee, Nokia Research Center, USA
Wolfgang Leister, Norsk Regnesentral (Norwegian Computing Center), Norway
Clement Leung, Hong Kong Baptist University, Hong Kong
Longzhuang Li, Texas A&M University-Corpus Christi, USA
Yaohang Li, Old Dominion University, USA
Jong Chern Lim, University College Dublin, Ireland
Lu Liu, University of Derby, UK
Damon Shing-Min Liu, National Chung Cheng University, Taiwan
Michael D. Logothetis, University of Patras, Greece
Malamati Louta, University of Western Macedonia, Greece
Maode Ma, Nanyang Technological University, Singapore
Elsa María Macías López, University of Las Palmas de Gran Canaria, Spain
Olaf Maennel, Loughborough University, UK
Zoubir Mammeri, IRIT - Paul Sabatier University - Toulouse, France
Yong Man, KAIST (Korea advanced Institute of Science and Technology), South Korea
Sathiamoorthy Manoharan, University of Auckland, New Zealand
Chengying Mao, Jiangxi University of Finance and Economics, China
Brandeis H. Marshall, Purdue University, USA
Sergio Martín Gutiérrez, UNED-Spanish University for Distance Education, Spain
Constandinos Mavromoustakis, University of Nicosia, Cyprus
Shawn McKee, University of Michigan, USA
Stephanie Meerkamm, Siemens AG in Erlangen, Germany
Kalogiannakis Michail, University of Crete, Greece
Peter Mikulecky, University of Hradec Kralove, Czech Republic
Moeiz Miraoui, Université du Québec/École de Technologie Supérieure - Montréal, Canada
Shahab Mokarizadeh, Royal Institute of Technology (KTH) - Stockholm, Sweden
Mario Montagud Climent, Polytechnic University of Valencia (UPV), Spain
Stefano Montanelli, Università degli Studi di Milano, Italy
Julius Müller, TU- Berlin, Germany
Juan Pedro Muñoz-Gea, Universidad Politécnica de Cartagena, Spain
Krishna Murthy, Global IT Solutions at Quintiles - Raleigh, USA
Alex Ng, University of Ballarat, Australia
Christopher Nguyen, Intel Corp, USA
Petros Nicopolitidis, Aristotle University of Thessaloniki, Greece
Carlo Nocentini, Università degli Studi di Firenze, Italy
Federica Paganelli, CNIT - Unit of Research at the University of Florence, Italy
Carlos E. Palau, Universidad Politecnica de Valencia, Spain
Matteo Palmonari, University of Milan-Bicocca, Italy
Ignazio Passero, University of Salerno, Italy
Serena Pastore, INAF - Astronomical Observatory of Padova, Italy
Fredrik Paulsson, Umeå University, Sweden
Rubem Pereira, Liverpool John Moores University, UK
Yulia Ponomarchuk, Far Eastern State Transport University, Russia
Jari Porras, Lappeenranta University of Technology, Finland

Neeli R. Prasad, Aalborg University, Denmark
Drogkaris Prokopios, University of the Aegean, Greece
Emanuel Puschita, Technical University of Cluj-Napoca, Romania
Lucia Rapanotti, The Open University, UK
Gianluca Reali, Università degli Studi di Perugia, Italy
Jelena Revzina, Transport and Telecommunication Institute, Latvia
Karim Mohammed Rezaul, Glyndwr University, UK
Leon Reznik, Rochester Institute of Technology, USA
Simon Pietro Romano, University of Napoli Federico II, Italy
Michele Ruta, Technical University of Bari, Italy
Jorge Sá Silva, University of Coimbra, Portugal
Sébastien Salva, University of Auvergne, France
Ahmad Tajuddin Samsudin, Telekom Malaysia Research & Development, Malaysia
Josemaria Malgosa Sanahuja, Polytechnic University of Cartagena, Spain
Luis Enrique Sánchez Crespo, Sicaman Nuevas Tecnologías / University of Castilla-La Mancha, Spain
Paul Sant, University of Bedfordshire, UK
Brahmananda Sapkota, University of Twente, The Netherlands
Alberto Schaeffer-Filho, Lancaster University, UK
Peter Schartner, Klagenfurt University, System Security Group, Austria
Rainer Schmidt, Aalen University, Germany
Thomas C. Schmidt, HAW Hamburg, Germany
Zary Segall, Chair Professor, Royal Institute of Technology, Sweden
Dimitrios Serpanos, University of Patras and ISI/RC ATHENA, Greece
Jawwad A. Shamsi, FAST-National University of Computer and Emerging Sciences, Karachi, Pakistan
Michael Sheng, The University of Adelaide, Australia
Kazuhiko Shibuya, The Institute of Statistical Mathematics, Japan
Roman Y. Shtykh, Rakuten, Inc., Japan
Patrick Siarry, Université Paris 12 (LiSSi), France
Jose-Luis Sierra-Rodriguez, Complutense University of Madrid, Spain
Simone Silvestri, Sapienza University of Rome, Italy
Vasco N. G. J. Soares, Instituto de Telecomunicações / University of Beira Interior / Polytechnic Institute of Castelo Branco, Portugal
Radosveta Sokullu, Ege University, Turkey
José Soler, Technical University of Denmark, Denmark
Victor J. Sosa-Sosa, CINVESTAV-Tamaulipas, Mexico
Dora Souliou, National Technical University of Athens, Greece
João Paulo Sousa, Instituto Politécnico de Bragança, Portugal
Kostas Stamos, Computer Technology Institute & Press "Diophantus" / Technological Educational Institute of Patras, Greece
Cristian Stanciu, University Politehnica of Bucharest, Romania
Vladimir Stantchev, SRH University Berlin, Germany
Tim Strayer, Raytheon BBN Technologies, USA
Masashi Sugano, School of Knowledge and Information Systems, Osaka Prefecture University, Japan
Tae-Eung Sung, Korea Institute of Science and Technology Information (KISTI), Korea
Sayed Gholam Hassan Tabatabaei, Isfahan University of Technology, Iran
Yutaka Takahashi, Kyoto University, Japan
Yoshiaki Taniguchi, Kindai University, Japan
Nazif Cihan Tas, Siemens Corporation, Corporate Research and Technology, USA
Alessandro Testa, University of Naples "Federico II" / Institute of High Performance Computing and Networking (ICAR) of National Research Council (CNR), Italy
Stephanie Teufel, University of Fribourg, Switzerland
Parimala Thulasiraman, University of Manitoba, Canada
Pierre Tiako, Langston University, USA

International Journal on Advances in Internet Technology
Volume 11, Numbers 3 & 4, 2018

CONTENTS

ValueML: A Proposal for Representing Values Embedded within Web Resources

Elio Toppano
Dipartimento di Scienze Matematiche, Informatiche e Fisiche (DMIF)
Università di Udine,
Udine, Italy
e-mail: elio.toppano@uniud.it

Abstract— **Growing on recent research work in the fields of Value Sensitive Design, Design for Well-Being and Disclosive Computer Ethics, this paper focuses on values embedded in multimodal web resources during the design and development processes. They are inscribed within the artifact as symbolic meanings or as a built-in use consequence. We propose a preliminary version of a markup language, called ValueML, that may be used for value representation and annotation. After a brief review of various perspectives on the concept of value and relevant taxonomies and vocabularies, we discuss the syntax and semantics of ValueML together with examples of manual annotation of video commercials.**

Keywords-value; annotation; semantic web; markup languages.

I. INTRODUCTION

This paper addresses the issue of values embedded in web information resources, their representation and annotation. It is an elaboration and extension of previous work presented in the IARIA Sixth International Conference on Building and Exploring Web Based Environments [1].

We are interested in how ethical, political, aesthetic and cultural values (to name only a few types) are, intentionally or unintentionally, constructed and articulated during the design and development of a product; how they are actually inscribed within the artifact and materialized through various semiotic resources (e.g., written and spoken texts, still and moving images, music and sound effects); how the values can be used for resource annotation and exploited in content retrieval, filtering, repurposing and reuse.

The need for value annotation of information resources is present in several application domains. In the fields of Marketing and Brand Communication, for instance, values constitute an important component of websites, commercial videos and advergames. They may refer to the advertised good or service (i.e., a value proposition) or, more generally, to a company's brand identity (i.e., the brand core values) and brand world (i.e., the brand world ethos).

Political parties, religious communities, non-profit organizations as well as social activists focus on values as one of the fundamental content and theme of their messages in designing websites and blogs. The project Values at Play (VAP), for example, is an initiative aimed at investigating how social and political values can be intentionally embodied in a digital game's architecture, interaction paradigm, and mechanics [2].

In the same vein, Value Sensitive Design [3], Value Centered Design [4], Design for Subjective Well-Being [5], Disclosive Computer Ethics [6] and Design for Sustainability [7] explore conceptualizations and methods for consciously addressing human values and related ethical concerns during the analysis and design processes of information systems.

All these initiatives assume that artifacts (and technologies, in general) are not morally neutral and that it is possible to identify *tendencies* in them to promote or demote particular values, and norms [6]. Such tendencies are embedded in the artifact in the sense that they can be identified and studied largely or wholly independently of actual uses of the artifact, although they manifest themselves in a variety of uses of the system (not necessarily in all uses!).

In addition, values are an important component of national, organizational and professional cultures. Therefore, independently of explicit design intentions, values are inevitably inscribed within information resources as a reflection of the culture of their clients, designers and developers (Culture *in* Design). Furthermore, communicative artifacts may be intentionally designed to adapt to the culture of target users (Design *for* Culture). This is at the base of the localization of web resources, a challenging issue addressed by several approaches in the field of cross-cultural design. The explicit representation and annotation of values present advantages in both cases: on the one hand, it forces the designers and developers to reflect on the intended or unintended values and the way they are embedded in their products; on the other hand, value annotation facilitates the identification and selection of those resources that are more appropriate for a target culture.

In spite of the growing interest on values manifested by several research studies, there is not yet a markup language and a value ontology or taxonomy that can be used for annotation of textual documents and multimodal resources on the web. This is a little disappointing considering the efforts that have been made, in the past, to model other experiential concepts such as pleasure, emotions, opinions and sentiments [8] [9]. The present work is a preliminary proposal aimed at filling this gap.

The paper is organized as follows. In Section II we discuss the concept of value from different perspectives and we illustrate available taxonomies and vocabularies of values. Next, in Section III we state the scope and the aim of the study. The main requirements of a language for value

annotation are, then, introduced in Section IV together with a possible solution, i.e., the ValueML. Section V is devoted to illustrate an example of annotation in the domain of video advertising. To this end, two paradigmatic cases have been selected among those used in the experimental activity made, so far. Section VI discusses some limitations and problems emerged during the assessment process of the current version of the language. Finally, in Section VII, some conclusions highlighting the benefits of value annotation are reported.

II. STATE OF THE ART

A. Defining value

The concept of value has several meanings according to the specific perspective from which it is considered. Looking at existing literature [10] [11] [12] [13] [14] [15], the term "value" has been interpreted variously as:

- an enduring belief that a specific mode of conduct or end-state of existence is personally or socially preferable to an opposite or converse mode of conduct or end state of existence (i.e., value as enduring belief system);
- the monetary sacrifice people are willing to make for a product (i.e., value as exchange);
- the utility of the physical properties of the product, which is realized only upon its use (i.e., value as perceived utility);
- an indicator of how much one desires a product or fears of loosing it (i.e., value as attachment);
- an index of social status, lifestyle, modernity (i.e., value as sign or meaning);
- an indicator of how the interaction with a product is aesthetically, cognitively or affectively worth to be made (i.e., value as good experience).

In his Value Theory [13], Schwartz, defines values as "desirable, trans-situational goals, varying in importance, that serve as guiding principles in people's lives". Most importantly, he identifies five main features of the conception of value that are implicit in the works of many theorists and researchers:

- values are beliefs tied inextricably to emotions;
- values are a motivational construct. They refer to desirable goals (e.g., idealized qualities or conditions in the world) people find good and strive to attain;
- values transcend specific actions and situations;
- values serve as standards or criteria to guide selection or evaluation of action, policies, people or events;
- values are ordered by importance relative to one another.

One aspect worth highlighting is the relationship existing between values and behavior. As claimed by Rokeach: "... values are determinants of virtually all kinds of behavior that could be called social behavior or social action, attitudes and ideologies, evaluation, moral judgment and justification of self to others, and attempts to influence others" [15].

Given the polysemy of the term value, one critical problem when using this concept for content analysis is to decide which interpretation is relevant for the purpose at hand. In addition, there is the need to disambiguate among different concepts that are in some way correlated such as values, needs, desires, preferences, and goals (see for example [10]). For some scholars values are abstract, desirable trans-situational goals; for others they are relatively stable individual preferences that reflect socialization; yet others consider values as cognitive representation of needs.

Another issue regards the classification of values. In [16] Brey illustrates an articulation of axiology - a branch of philosophy concerned with a general analysis of value - that provides structure and overview to relevant values belonging to traditional theories including cultural values of Theories of Good applied to new media. Figure 1 schematizes this classification.

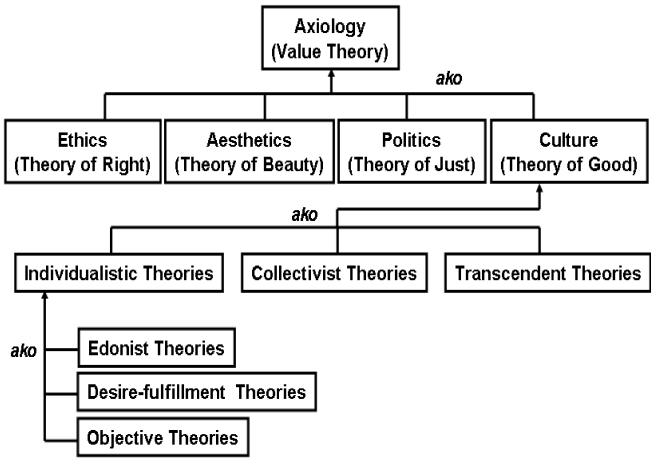

Figure 1. A classification of values according to Brey [16].

As shown in the figure, axiology can be decomposed into four main fields each one devoted to a specific type of value: Ethics concerning with the *right* (i.e., the understanding of what kinds of actions are right and, therefore, obligatory and which ones are wrong and, therefore, impermissible); Aesthetics concerning with the *beautiful*; normative Politics concerning with the *just* (i.e., the understanding of how the state ought to operate in relation to its citizens and how it should distribute powers and goods); and Theories of Good concerning with the sort of things in life that are considered *good* and, thus, worth striving for such as well-being, happiness and personal flourishing (i.e., eudaimonia). In particular the individualistic study of well-being has yielded three major types of theories namely hedonist (well-being consists of the presence of pleasure and the absence of pain), desire-fulfillment (well-being lies in the fulfillment of one's desires) and objective theories (well-being is the result of objective conditions of persons such as liberty, friendship, autonomy, wisdom, etc.). Collectivist theories hold that the greatest good is not the good of individual human beings but the good of a community or society at large; Transcendent theories point to one or more transcendent state-of-affairs or qualities that are held to constitute the highest good such as

the glory of God, the natural order of things, eco-systemic integrity, knowledge or information.

B. Value inventories and taxonomies

What emerges from research literature is that a unified theory or model of values currently does not exist. Available proposals differ along several dimensions, including:

- *representational approach*: the simplest and most wide-spread approach is the *categorical* approach that uses a word/label to represent a value. An alternative approach is the *dimensional* one that tries to capture the essential properties of values, by positioning them on a continuous space spanned by two or more dimensions. Holbrook, for example, proposes intrinsic-extrinsic, self oriented-other oriented, and active-reactive, as the main dimensions for classifying values [17];
- *purpose*: most of the proposals have been designed for survey research; only few have been applied to content analysis [14];
- *scope of application*: proposals span a wide range of scopes and application domains. Some proposals regard personal or individual values; some represent the values of groups of people, communities of practice or interests, institutions or companies, others focus on national values. Domains include psychology and social research; professional ethics (e.g., the values that guide the day-to-day activities of business managers), marketing and advertising; design, technology, and more;
- *number and type of distinct values*: proposals widely vary in the number of specific values (or dimensions) that are considered relevant;
- *structural organization of values*: most proposals are unstructured lists or inventories of items representing different types of values; others present hierarchical structures or taxonomies. Some proposals distinguish between instrumental and ends values; some organize values according to similarity/affinity or opposition/contrast relationships. Yet other proposals use part/whole relations (i.e., mereologies) to represent both elementary and aggregate (or composite) values;
- *conceptual definitions*: few proposals provide a clear and accurate definition of value categories or dimensions. Therefore, there are ambiguities (i.e., multiple interpretations of values), and "semantic confusion" both within and across the various proposals;
- *terminological realizations*: proposals use different vocabularies for denoting value categories. In some cases, the same term/label is used, within different vocabularies, for representing different value concepts (a case of homonyms). As an instance, the term "autonomy" is defined, within Value Sensitive Design [18], as: "people ability to decide, plan, and act in ways that they believe will help them to achieve their goals". The very same term in [19] is used to denote: "feeling that one's activity is self-chosen and self-endorsed". In other cases, different terms are used to denote the same value concept (a case of synonymy). What is termed "autonomy" in one vocabulary [19] is denoted by "self-direction" in another proposal [13]. Moreover, some terms (e.g., autonomy, self actualization) are common to various proposals while others (e.g., informed consent, humor) are present only in some specific vocabulary;
- *homogeneity/heterogeneity*: some inventories represent homogeneous values; others merge values having different nature such as hedonic values (e.g., pleasure) with ethical (e.g., morality, virtue), political (e.g., justice), and cultural values (e.g., life quality, happiness) or different generality or aggregation levels. Some examples are as following.

Schwartz proposed a set of fifty-six human values that are organized into ten basic categories: self-direction, stimulation, hedonism, achievement, power, security, conformity, tradition, benevolence, and universalism [13]. These categories constitute a circular pattern where congruent values (e.g., achievement and power) are located on adjacent positions while conflicting values (e.g., achievement and benevolence) on opposite sides. At a more general level, basic categories are aggregated into four macro-categories, namely, openness to change, self-transcendence, conservation and self-enhancement.

Boztepe, focusing on user's values, proposed a classification including nineteen different values organized into four main categories namely, utility, social significance, emotional, and spiritual [11].

Value Sensitive Design uses a flat list of values to be used in computer systems. Examples are: privacy, freedom from bias, informed consent, accountability, property rights, to name only a few [3].

Specific taxonomies have been proposed in marketing [20] and in game design [21]. Floch, for example, distinguishes four types of product values - practical, critical, utopian, and ludic values - that are at the base of the main marketing strategies. Flanagan et al. use a set of seventeen values (e.g., diversity, justice, inclusion, equality, environmentalism, creativity, trust, etc.) for the analysis and design of digital games. Recent research in Positive Design focuses on hedonic values (e.g., pleasure) and eudaimonia (e.g., personal flourishing) [5].

It should be clear, from the above discussion, that it is difficult to select a proposal as a common reference standard or to create a shared standard by combining elements from existing vocabularies. Ideally, for content annotation in a given application domain, an adequate *value ontology* (i.e., conceptualization and vocabulary) should be manageable, that is, it should contain the minimum number of relevant value categories for the task at hand, and minimize conceptual ambiguity and redundancy by making each value concept unique and independent from the other concepts. For example, concepts such as "achievement" and "success" may be ambiguous when they are in the same inventory. However, by synthesizing them under the concept of "accomplishment" may avoid the ambiguity. A prerequisite

is that values do not remain underdeveloped: their nature and meaning must be clearly defined and distinguished. This is an important step for applications in the Semantic Web field. As a final remark it should be noted that a value inventory or taxonomy not only displays what values categories are available for analysis but also provides a descriptive tool for researchers to focus their discussion about values.

III. SCOPE AND AIM OF THE STUDY

A. The values inscribed within a web resource

Figure 2 shows the general framework we have adopted for communication-based design [22]. It accounts for the real sender and receiver. They are connected by means of an instrumental medium - a device such as a PC, a tablet or a smart phone and a channel such the Internet - that enables the circulation of a web resource. By the term *web resource* we mean an information object (e.g., a message) that is realized by at least one accessible web representation and encoded in a media type.

The sender is involved in the design and realization of the web resource. It is usually a team including, for example, the designers and developers as well as the client, sponsor and firm/company. The interaction between the sender and the resource takes place within a production context involving space-time location, the organizational and socio-cultural environments with roles, norms, policies, standards and technologies.

The receiver, an individual or a group, interacts with the resource within a context of use that may, or not, be synchronized with the production one as in the case of different cultural contexts. In this framework the web resource plays the role of a *mediator* between the intentions of the sender and the interpretation of the receiver.

We shall focus on the production context of the above framework: we are interested in the values that are inscribed (or embodied) within the resource as a kind of meaning during its design and development. We call these values the "values *in* the product" to distinguish them from other kinds of values such as, for example, the economic value of the product (i.e., its exchange value), the value that a resource gains passing through a chain of activities (i.e., the value chain) or the value of the product as experienced by a user during its use (i.e., the perceived use value). These interpretations can be referred to as the "value *of* the product".

The values in the product have different sources. Some values are strictly related to the stakeholders (the firm, the designers, or the users). These include, for example, the values that are associated to the personality of a brand, its identity, the brand world's ethos or the assumed values of the target user group to whom the message is directed. These values are usually explicitly expressed by a set of non-functional requirements. As an instance: to design a video commercial or a website to promote the core values of a company or to resonate with the cultural values of a specific community of practice or interest.

Other values (called collateral values) are, intentionally or unintentionally, embedded within the web resource as the result of specific design or implementation decisions. It should be noted that a design solution is always under-determined by specific functional requirements leaving to designers and developers numerous open-ended alternatives as they proceed through the process, including some that implicate values [23].

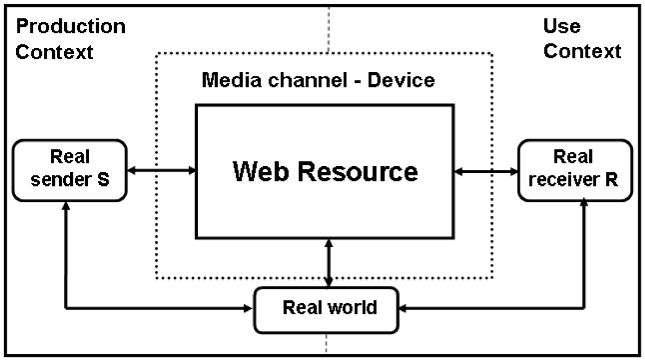

Figure 2. The communication-based design framework adopted in the study.

The selection of an alternative solution with respect to another is guided by criteria that are often value-laden and reflect the personal or professional beliefs, preferences, cultural milieu of these persons.

In the case of information resources, design decisions may refer to several aspects such as, for example, i) the adopted conceptual model or metamodel of the resource; ii) the articulation of content meaning; iii) the kind of semiotic media used for meaning presentation or expression, iv) stylistic choices (e.g., space-time layout of information, colors, typography); (v) the specific technologies employed during the implementation of a design solution, and vi) the definition of intended use in the destination context.

Here are some examples. The adoption of the Semantic Markup for Web Services (OWL-S) instead of the Web Service Modeling Ontology (WSMO) reflects different values and has different ethical implications as discussed in [24].

In a narrative commercial clip, ethical or aesthetic values can be inscribed in the story (content) or in the visual and auditory qualities (expression); in a digital game they can be embodied in game mechanics (e.g., in game elements, actions, rules, reward system), game dynamics (e.g., player's point of view, course of action) or experience (e.g., game atmosphere, narrative theme, expressed or induced emotions).

According to [6], it can be useful to distinguish the following two main cases:

- an information resource may be expressive of values (i.e., *expressive conception* of embedded values) in that it contains symbolic meanings that refer to values. These values may represent the values of designers, clients or users;
- an information resource may have embedded values understood as special kind of built-in consequences. This conception (i.e., *causalist conception* of

embedded values) relates values to causal capacities of the resource to affect the environment. In other words, the resource use causes a state of the world that realizes some kind of value. In Persuasive Technologies [25], Design for Sustainable Behavior [7], and in applications inspired to Nudge Theory [26], values are directly related to the intended behavior or state we want to be enabled, induced or fostered in users.

Of course, the fact that a resource is expressive of values does not imply that it also functions to realize these values. Whether this happens or not remains an open question.

B. The problem addressed: value analysis and annotation

Generally speaking, value taxonomies and vocabularies can be exploited in four different use cases:

- manual annotation of multimodal resources with inscribed values. By *multimodality* we mean the use of two or more representational modalities (or semiotic "languages") to express the content of an information resource. An example is to identify and represent the values communicated by the visual (i.e., written text, images) and auditory (i.e., spoken language, music, effects) tracks of a video commercial;
- automatic value detection and classification. The goal, here, is to model the means-ends relationships existing between measurable features of multimodal artifacts and abstract constructs such as value concepts. An example is to infer the values that are at the base of behaviors or situations depicted by an image; or to infer the values that produce certain emotions expressed by the characters of a story;
- value generation, that is, simulation of specific values by an appropriate selection and composition of multimedia content and expression. An example is to select the actions of the characters of a story that best communicate some specific values;
- value assessment, that is, the use of value inventories for empirical tests aimed at measuring how people would rank or rate the relative importance of items in the given list of values. An example is to profile a group of individuals to adapt the design of a new resource to their values.

An important aspect that is exploited in automatic detection and generation is that values have been acknowledged as a key predictor and explanatory factor in investigating human and social dynamics.

Our study focuses on the first use case. The problem we intend to address is, thus, the following: to design a general-purpose language for the manual annotation of values inscribed within a multimodal resource.

The language should let the annotator to define the scope of a value annotation and to describe the value itself by referring to a specific and shared vocabulary. Notice that, the focus is on message production rather than use. This is not to deny that annotations made by the users are important. Simply, social tagging comes after the product has been developed and published and has different goals. It may be used, for example, to assess the effectiveness of intended value communication.

We envisage several possible ways in which the annotation could be used including:

- retrieval and selection/filtering of resources or part of them on the base of intended embodied values. It may be possible, for example, to annotate specific fragments of a multimodal resource with intended values and then retrieve the fragments using the values as keywords;
- reuse of a resource for new goals or contexts (i.e., repurposing). The identification and value annotation of multimodal fragments enables a designer to reuse the content of the fragment for new goals/objectives or in new communication contexts;
- exploitation of design knowledge embodied within an artifact for new products. Linking values to fragments is a way to explicitly represent how values are communicated in *that* artifact. This knowledge is design knowledge that may be used as inspirational for innovative products and design solutions;
- construction of a shared data base of value annotation resources that can be used as a ground base or training set for automatic recognition of values or for scientific research.

IV. THE VALUEML LANGUAGE

A. Language requirements

The following is an initial list of requirements for the development of a language for value annotation. They are based on an understanding of the needs arising from concrete scenarios of manual annotation of multimodal texts. We have structured requirements according to a specific model of annotation that distinguishes among: 1) *annotated data* (i.e., the subject or target of the annotation), 2) the *annotating data* (i.e., the object or body of the annotation), 3) the *annotation relation* (i.e., a predicate that defines the type of relationship existing between annotated and annotating data) and 4) the *context* in which the annotation is made [27]:

- *Annotated data*. The desired language should allow the annotator to associate values to an entire resource or to specific parts of it at different aggregation levels. This requires a mechanism for addressing spatiotemporal segments of the resource (e.g., a part of an image, text section, video or audio segment). In addition, it should be possible to annotate different semiotic modalities (e.g., written texts, spoken words, still images, music, videos). Values can be communicated in sequence or in parallel (i.e., simultaneously); within a single modality or multiple modalities so the language is requested to provide means to address all these possibilities.
- *Annotating data*. The desired language should allow the annotator to use a wide range of value types (i.e., categories) and taxonomies. We are interested in moral, ethical, political, aesthetic, and cultural values. As discussed in the previous sections any attempt to standardize the description of values using

a finite set of fixed descriptors is doomed to failure: even scientists cannot agree on the numbers of relevant values, or on the names that should be given to them. Given this lack of agreement on value descriptors and classification, the only practical way to tackle this problem is to allow annotators to "plug in" vocabularies that they consider appropriate for their applications. As a consequence, the language should be sufficiently flexible and modular so that the appropriate vocabulary of descriptors for the target use can be chosen. The aim is also to make scientific concepts of values practically applicable. Available alternatives that have been developed in literature are not sufficiently known, so people generally use a restricted set of not well defined values without asking themselves if they are the most appropriate choices for the application at hand.

- *Annotation relation*. It should be possible to explicitly represent the functional role played by the object with respect to the target of annotation. Moreover, the language should allow the annotator to represent means-ends relationships linking values to their material realization in the given resource. A means-ends chain can be followed in two directions: top down from a value to specific resource features in order to understand *how* the value has been materialized in the resource; bottom up, from specific features to values in order to understand *why* the resource has been design the way it is. This information can be used to unfold design knowledge embodied in a given resource. In addition, if values are viewed as a kind of meaning then they may be dispersed across content (e.g., narratives, denotative and connotative meanings) as well as expression (e.g., visual and auditory qualities). As an example, brand values and brand ethos could be communicated by a storyline while aesthetic values by the product look and feel.
- *Context*. The language should allow the representation of contextual information including: i) general metadata regarding the type of annotated resource, its location, the author of the annotation, the intention and the method of annotation, etc. ii) a measure of relevance (importance) of a given value with respect to the other values within the same resource; iii) a measure of confidence in the accuracy of the annotation. It should be noted that the expression of a value in an informational resource may be masked by another one, it may be inhibited, minimized or even exaggerated. Therefore, the human annotator needs to indicate the degree of importance and confidence that a certain attribution is correct.

Additional requirements regard: i) a compatibility with the broad range of existing value theories; ii) the interoperability of the desired language, i.e., the possibility to embed it in other markup languages (e.g., EmotionML, SMIL); iii) extensibility, i.e., the possibility to extend the language in order to satisfy future use cases and thus avoid rapid obsolescence. The following section describes the main features of a draft language proposal intended to satisfy most of the above design requirements.

B. Language syntax and semantic

In developing the ValueML language we take inspiration to the recent W3C initiative of EmotionML for the annotation of emotions [8]. ValueML may be seen as a complementary tool for describing experiential and socio-cultural aspects of web resources. The language uses an XML-based syntax. Table I illustrates the main elements of a document structure. The root element of a standalone ValueML document must be `<valueml>`. It may contain a single `<info>` element, providing metadata regarding the annotated information resource such as, for example, its media type, name, description, or web location. The `<info>` element may be followed by the declaration of a controlled vocabulary specifying the set of value labels (or categories) that can be used in the annotation. We draw on existing literature to propose the set of value categories. One or more value annotations may follow that are self-contained within the `<value>` element. One of the envisaged uses of ValueML is to be used in the context of other markup languages. In such cases, there will be no `<valueml>` root element, but `<value>` elements will be used directly in other markup. In order to address these cases the `<value>` element has been provided with the attribute "`vocabulary-set`" for specifying the URI of the external vocabulary to be used. Each value annotation includes two children elements namely the `<category>` and the `<reference>`. The `<category>` element has three attributes (see Table II). The attribute "`name`" is used to specify a value. Its content must be an item of a controlled vocabulary (internal or external to the document).

The attribute "`relevance`" is used to express the relative importance of the value with respect to other values embodied within the resource. This information is described by a floating-point number in the close interval [0,1]. Finally, the attribute "`confidence`" is used by the annotator to express his or her confidence that the annotating value is correct. Again, this information is described by a floating-point number in the close interval [0,1].

Similarly, the `<reference>` element has three attributes (see Table III). The attribute "`uri`" is used to refer to the annotated resource or part of it. Its content must be a Media Fragment URI [28] consisting in four parts:

```
<scheme name>:<hierarchical part>
[?<query>] [#<fragment>]
```

There are, therefore, two possibilities for representing the media fragment addressing URIs: the URI query part, or the URI fragment part. For dynamic resources such as sound objects or videos, segments are represented by time intervals. An interval is denoted by specifying begin and end times, e.g., t=t1,t2 where ti is expressed in seconds and milliseconds according to NPT (Normal Play Time). For static resources such as images, segments are represented by regions. A region is denoted by a rectangular selection, e.g.,

xywh, where x,y represent the top, left corner of the selection and w and h denote width and high, respectively.

The attribute "`role`" specifies the type of relationship existing between the annotated and annotating information. Currently, we suggest two possible instances: a value *is signified by* a resource or a value *is expressed by* the resource. The rationale is that we adopt a semiotic stance according to which an information object can be viewed as a structured system of signs constituted by a content layer (the signified) and an expression layer (the signifier).

Finally, the attribute "`modality`" can be used to describe *how* the value is communicated. It allows the annotator to specify the representational modality used for value communication (e.g., written text, sound objects, images or videos) and the specific features that are used for the task. As an instance, an image can be read in terms of plastic features (i.e., shapes, colors, textures, positions, orientations, sizes) or in terms of figurative elements (i.e., the objects, persons, actions, and scenes depicted in the image). Both kinds of readings may by used to convey values. Currently, we have identified a limited set of possible instances of the "modality" attribute including: by *storylineContent*, by *imageExpression*, by *imageContent,* by *soundObjectContent*, to name but a few. Tables II and III summarize the main features of the above discussion. Table IV illustrates the main elements used by ValueML to define a value vocabulary as a list of items representing value categories. The `<info>` element can be used to specify metadata associated to the vocabulary, e.g., the scope and purpose of the inventory. Currently, the type of vocabulary must be a categorical one.

V. CASE STUDIES

In order to assess the feasibility of value annotation with ValueML in concrete cases we focused on annotation of video commercials. Video commercials are a particular subclass of audio-visual resources that are characterized by specific properties in terms of: purpose, form, editorial context and use. Generally speaking, video commercials are designed to promote goods, services, ideas, brands or persons (e.g., politicians). This can be done by using several forms of appeal (i.e., logos, ethos and pathos) and different rhetorical strategies including analytical (e.g., logical argumentation) and narrative processing (e.g., storytelling) [29]. In particular, it has been shown that narrative advertising is particularly effective in persuasion (e.g., belief change) due to the transportation effect that has been studied and empirically demonstrated by [30].

To give an example of value annotation, we have selected two paradigmatic cases. The first case - the Citroen BX ad campaign - is a narrative video promoting a specific model of car by describing an experience of use of the good. The second case - the Thai Insurance ad campaign - is a narrative video that does not advertise a specific product or service but is aimed at communicating the core values of the Thai Insurance brand using a story. The following sections describe each case in detail. In both cases annotation is preceded by an analysis of the clips that is driven by a semiotic metamodel of the video as discussed in [27] [31]. The metamodel distinguishes four interrelated levels of analysis (Figure 3): i) the textual level representing the concrete/physical manifestation of the video content in terms of audio-visual features; ii) the discourse level referring to thematic, figurative, rhetorical aspects; iii) a shallow narrative level describing the story told by the video in terms of abstract roles (called actants) and narrative schemes (e.g., the narrative canonical scheme), and, iv) a deep narrative level that uses a specific tool called semiotic square to articulate deep semantic meanings such as narrative values (axiology). Signification unfolds by crossing these levels from shallow features of the video to the most abstract and deep ones. This metamodel is particular interesting because it explicitly represents the values inscribed in the product and relates these values to the storyline content and expression (i.e., sensorial qualities). Value annotation concerns the deep Semio-Narrative level.

A. The Citroen BX ad campaign

The clip selected for analysis and annotation represents a well-known ad campaign by Citroen [32]. It was produced in 1982 to advertise the BX model. The video lasts 47 s. It is constituted by a sequence of 18 shots. The narrative is based on a *valorization process*. By this term we intend a process by means of which an object/service (a car model in this case) is staged in a way that emphasizes (i.e., gives value to) some attributes of the object with respect to others according to specific consumer's values. Therefore, content annotation must be based on a classification of this kind of values. A possible choice is the model proposed by Floch [20].

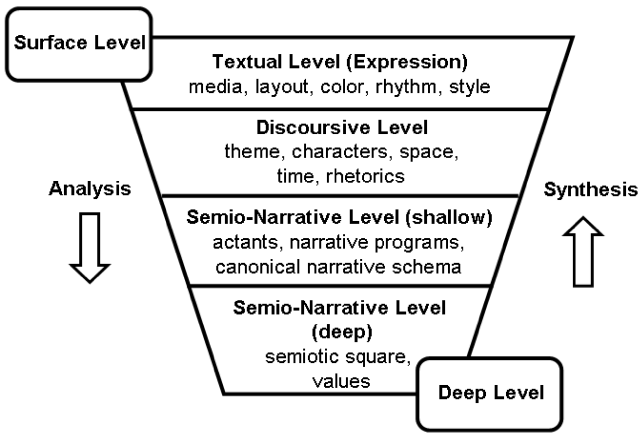

Figure 3. A schematic view of the metamodel used for the analysis of narrative video commercials.

In his model, consumer's values are classified into four classes namely, practical, critical, utopian and ludic values. They represent the vertices of a semiotic square (see the bottom of Figure 3). Briefly: practical values refer to utility, usefulness; critical values to convenience, performance, quality; utopian values to identity, reflection, social relations, and ludic values to surprise, madness, astonishment, irony and pleasure including aesthetic pleasure.

TABLE I. MAIN ELEMENTS OF A VALUEML DOCUMENT STRUCTURE.

Annotation	<valueml>
Definition	The root element of a ValueML document.
Children	The element may contain a single <info> element providing metadata about the annotated resource. It may contain one or more <vocabulary> elements. It may contain one or more <value> elements.
Attributes	Required: • version: indicates the version of the specification to be used in the document. Documents using this specification must use 1.0 for the value. • namespace: declaration for ValueML.
Occurence	This is the root element; it cannot occur as a child of any other ValueML element.

Annotation	<value>
Definition	This element represents a single value annotation.
Children	The element must contain a single <category> element followed by a <reference> element.
Attributes	Optional: • vocabulary-set: a URI indicating an external repository of value vocabularies and the specific vocabulary to be used. Note: this attribute is requires if no vocabulary is declared within the ValueML document.
Occurence	As a child of <valueml> element.

TABLE II. THE <CATEGORY> ELEMENT.

Annotation	<category>
Definition	Description of a value using a category name.
Children	None.
Attributes	Required: • name: indicates the name of a value category, which must be contained in the declared vocabulary. Optional: • relevance: specifies the importance of a value. It must be a floating-point number in the closed interval [0,1]. • confidence: the annotator's confidence that the annotation given for this category is correct. It must be a floating-point number in the closed interval [0,1].
Occurence	As a child of the <value> element.

TABLE III. THE <REFERENCE> ELEMENT.

Annotation	<reference>
Definition	This element is used to relate a value to the annotated resource or part of it.
Children	None.
Attributes	Required: • uri: a URI indicating the actual reference target. The URI may be extended by a media fragment or a media query. Optional: • role: specifies the type of relation between the value and the external item referred to. Its content must be one of "signified_by", "expressed_by". • modality: describes the modality through which a value is signified or expressed. Possible instances are "by_storylineContent", "by_imageExpression", "by_imageContent", "by_soundObjectContent", "by_soundObjectExpression". Note: the above list is an open set to allow for a more fine-grained distinctions.
Occurence	As a child of <value> element.

TABLE IV. THE <VOCABULARY> AND <ITEM> ELEMENTS.

Annotation	<vocabulary>
Definition	This element contains the definition of a value vocabulary.
Children	A <vocabulary> element must contain one or more <item> elements. It may contain a single <info> element providing metadata about the vocabulary itself.
Attributes	Required: • id: a unique vocabulary identifier. Optional: • type: must be "category".
Occurence	As a child of <valueml> element.

Annotation	<item>
Definition	This element represents the definition of one vocabulary item.
Children	None.
Attributes	Required: • name: a name for the item, used to refer to this item in the <category> element.
Occurence	As a child of <vocabulary> element. Note: a vocabulary must contain at least one <item> element.

In the considered video clip, values are communicated as follows:

- the first segment of the video (time interval: [0 s, 33 s]; 14 shots) represents practical values (see Figure 4). A red car leaves Paris at midnight (Minuit, Paris ...) under the rain. After 8 hours it gets to the sea (... 8 heures, la mer). The car is presented as a safe, confortable, and quick mean to escape from the everyday city life. Onboard a young lady, takes off her hat, smiling. An off screen voice (by Julien Clerc) sings: "J'aime, J'aime, J'aime";

Figure 4. Four key frames of the first segment of the video clip representing practical values.

Figure 5. Two key frames of the second segment of the video clip representing ludic values.

Figure 6. A key frame of the final segment of the video clip representing utopian values.

- a following segment (time interval: [34 s, 40 s]; 3 shots) is used to communicate ludic values (see Figure 5). The car suddenly dives in the sea without it could be possible to attribute this mad action to the driver that is never shown. The plunge, unexpected and irrational, represents the negation of practical values shown in the previous segment;
- a final segment (time interval: [41 s, 47 s]; one shot) represents utopian values (see Figure 6). Here, the car (Citroen BX) is no more an instrument, it is a subject, it lives (Citroen BX. Elle vit.).

Figure 7 shows the semiotic square used to represent the logical articulation of a basic opposition between practical and utopian valorizations. Arrows represent the trajectory of values expressed by the Citroen BX clip during presentation. Practical and ludic values are communicated through the visual track, while the utopian valorization is explicitly expressed by a voice over.

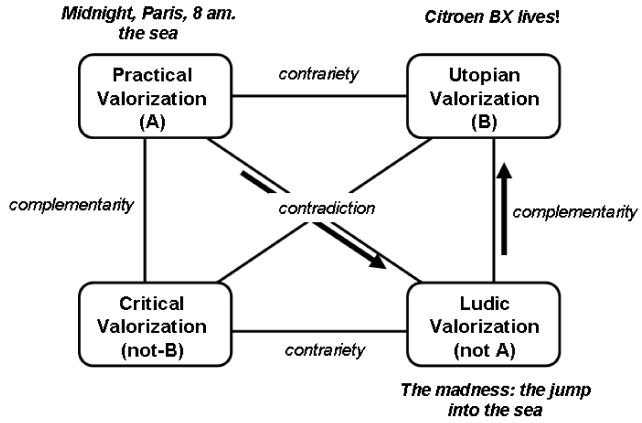

Figure 7. Semiotic square of consumer values

Figure 8 shows an example of annotation of the considered video clip using the ValueML.

Some observations are worth mentioning. First, the controlled vocabulary used for the annotation - the Floch's classification - has been defined within the document itself using the `<vocabulary>` element. The relevance of the annotating value has been assigned on the base of the time span of the video segment to which the value is associated. Therefore, values associated to longer video segments have a greater relevance with respect to values associated to shorter ones. This is a raw estimation of relevance that does not take into account other aspects such as, for example, the presence of multiple modalities communicating the same value within a segment.

Confidence is high. The rationale is that the discovery of values and the association of values to specific video segments is based on the availability of the original script and critical essays about the considered video that provide the main source of annotation knowledge [20]. Therefore, we consider the annotation as a correct annotation. Annotated segments are represented by query segments. This choice produces a new resource and allows for transcoding activities. Therefore, if a URI query addresses a single frame

out of a longer video it can be transcoded into a still image. This is not possible with URI fragments [28].

```
<valueml version="1.0" xmlns="http://www.example.com/2018/valueml"
xmlns:meta="http://www.example.com/metadata">
 <info>
  <meta:media-type> video </meta:media-type>
  <meta:media-name> citroen_bx.webm </meta:media-name>
  <meta:doc> Commercial clip by Citroen. The spot has
   been produced by RSCG in 1982. It is available at the
   URI:https://www.youtube.com/watch?v=jH2HLRpIq2Y </meta:doc>
 </info>
 <vocabulary id="Floch-semiotic-square">
   <item name="practical-valorization" />
   <item name="utopian-valorization" />
   <item name="critical-valorization" />
   <item name="ludic-valorization" />
 </vocabulary>
 <value>
  <category name="practical-valorization" relevance="0.7"
    confidence="0.9"/>
  <reference role="signified_by" uri="file:citroen_bx.webm?t=0,34"
    modality="by_storylineContent"/> </value>
 <value>
  <category name="ludic-valorization" relevance="0.2"
    confidence="0.9"/>
  <reference  role="signified_by" uri="file:citroen_bx.webm?t=34,41"
    modality="by_storylineContent"/> </value>
 <value>
  <category name="utopian-valorization" relevance="0.1"
    confidence="0.9"/>
  <reference  role="signified_by" uri="file:citroen_bx.webm?t=41,47"
    modality="by_soundObiectContent"/> </value>
</valueml>
```

Figure 8. Annotation of the BX video clip with ValueML.

B. *Thai Life Insurance ad campaign*

The video commercial [33] represents a snapshot of the daily life of a young man, his encounters and emotional experiences. It lasts 3 minutes and it is constituted by a sequence of 76 shots. Because there is not a specific object to be promoted, we refer here, for the annotation, to the classification of universal values proposed by Schwartz [13]. Figure 9 shows the declaration of the relative vocabulary using ValueML. This example is aimed at showing how the markup language can be used to define an external repository of value vocabularies.

One problem we encountered during the analysis of the Thai clip regards how to assign values to the events of the story in a way that could be sufficiently "objective" and reproducible. The idea was to exploit the link existing between values and behaviors. To this end, we used the Portrait Values Questionnaire (hereafter, PVQ) developed by Schwartz for the measurement of users' values [34]. The PVQ uses a list of forty *portraits*, each one describing what is important for a hypothesized male or female individual and what he/she likes. For the sake of clarity the following are four examples of portraits (the numbers reflect their position in the original PVQ list):

P-1: thinking up new ideas, and being creative is important to him. He likes to do things in his own original way;

P-2: it is important to him to be rich. He wants to have a lot of money and expensive things.

P-12: it is very important to him to help people around him. He wants to care for their well-being;

P-19: he strongly believes that people should care for nature. Looking after the environment is important for him.

```
<valueml version="1.0" xmlns="http://www.example.org/2018/valueml">
 ...
 <vocabulary id="Floch-semiotic-square">
   <item name="practical-valorization" />
   <item name="utopian-valorization" />
   <item name="critical-valorization" />
   <item name="ludic-valorization" />
 </vocabulary>
 <vocabulary id= "Schwartz-value_types">
  <info> .. general metadata.. </info>
   <item name="power" />
   <item name="achievement" />
   <item name="hedonism" />
   <item name="stimulation" />
   <item name="self-direction" />
   <item name="universalism" />
   <item name="benevolence" />
   <item name="tradition" />
   <item name="conformity" />
   <item name="security" />
 </vocabulary>
 ...
</valueml>
```

Figure 9. Definition of a repository of value vocabularies with ValueML.

In value surveys, the PVQ is used by asking users to mark the portraits that better reflect themselves. An algorithm is then executed to map answers with value categories. As an instance, portrait P-1 above is associated to self direction; P-2 to power, P-12 to benevolence, and P-19 to universalism. We re-purpose the tool for content analysis by using it to characterize the behavior of characters in a narrative. In other words, instead of using the PVQ for the analysis of users we employ it for the analysis of the main characters in narratives. The method we have envisaged/foreseen for this task can be summarized as follows:

Step 1: decompose the video into its constituent shots, then aggregate shots into scenes on the base of the unity of intent or action/behavior of the characters represented in the shots;

Step 2: use the PVQ list of portraits to characterize each scene obtained in the previous step. This step exploits the strong association existing between portraits and expected behaviors. The annotator is assumed to perform an abduction process to infer a portrait on the base of observed behaviors in the video;

Step 3: map portraits and relative scenes with their associated values according to the PVQ algorithm;

Step 4: evaluate the relevance of each value by computing the total time of occurrence of the value (i.e., the sum of the time duration of all the scenes/shots in which the related behaviors are presented) with respect to the duration of the whole clip.

The application of the method has been supported by the use of the ELAN tool [35]. Results indicate that three main values are inscribed in the commercial video namely universalism (relevance: 19%), benevolence (relevance: 38%) and tradition (relevance: 12%). Figure 10 shows key

frames extracted from scenes showing behaviors associated to these values.

Figure 11 illustrates a fragment of the video annotation using ValueML.

Figure 10. Key frames obtained from scenes that represent specific values: universalism (top frames), benevolence (middle frames) and tradition (bottom frames).

Some observations are worth mentioning. First, the definition of an external repository of vocabularies (see Figure 9) allows the annotator to potentially use different vocabularies in the same document. Notice that a URI fragment is used to specify a vocabulary as the content of the "`vocabulary-set`" attribute in each `<value>` element. Second, each value is communicated by one or more portraits that are spread over several scenes. Benevolence, for example, is associated to five different portraits that are well represented in several scenes/shots of the video. These portraits are:

P-3: he thinks it is important that every person in the world be treated equally. He believes everyone should have equal opportunities in life;

P-19: he strongly believes that people should care for nature. Looking after the environment is important to him;

P-23: he believes all the worlds' people should live in harmony. Promoting peace among all groups in the world is important to him;

P-29: He wants everyone to be treated justly, even people he doesn't know. It is important to him to protect the weak in society;

P-40: it is important to him to adapt to nature and to fit into it. He believes that people should not change nature.

The main consequence is that a value category is associated to more than one reference element in the annotation.

Third, the modality attribute can be further specialized by taking into account the specific component of a storyline content that is used, in the clip, to express the value (in this case the main character's behavior).

Another observation concerns the audio track of the video. This is composed by a combination of music, effects and voice over. The voice over contributes to the communication of the values but in a very different way: it does not express the intended value by a description of the associated behavior but through the negation of the opposite behavior (or portrait).

```
<valueml version="1.0" xmlns="http://www.example.com/2018/valueml"
xmlns:meta="http://www.example.com/metadata">
    ...
<value
vocabulary-set="http://www.example.com/2018/vocabularies.xml#Schwartz-
value_types" >
  <category name="benevolence" relevance= "0.38"
    confidence="0.7"/>
  <reference role="signified_by" uri="file:thai_ins.mp4?t=11,15&t=45,8,52,2"
    modality=" by_characterBehaviour"/> </value>
<value
vocabulary-set="http://www.example.com/2018/vocabularies.xml#Schwartz-
value_types" >
  <category name="benevolence" relevance= "0.38"
    confidence="0.7"/>
  <reference role="signified_by" uri="file:thai_ins.mp4?t=22,7,32,4"
    modality=" by_characterBehaviour"/> </value>
    ...
<value
vocabulary-set="http://www.example.com/2018/vocabularies.xml#Schwartz-
value_types" >
  <category name="universalism" relevance= "0.19"
    confidence="0.7"/>
  <reference  role="signified_by" uri="file:thai_ins.mp4?t=5,8,8"
    modality=" by_characterBehaviour"/> </value>
    ...
</valueml>
```

Figure 11. Partial example of annotation of the Thai clip.

As an instance, the voice over of the Thai clip says :"He does not want to be rich" referring to the character of the story. This expresses an opposite value with respect to power (a value associated to the portrait P3: he wants to be rich). Looking at the Schwartz circle of values the opposite of power is universalism that is coherent to what is communicated by the visual track. The problem of coherence of meaning expressed by different modalities is a critical issue during the design of a multimedia and a useful source of redundancy for annotators during analysis and annotation.

Another dimension of coherence concerns the degree of localization of the video commercial. Are the values embodied in the clip adapted to the Thai Culture? If we refer to the characterization of national cultures made by Hofstede's theory [36] then the answer is definitely yes. Thailand is considered a nation that scores very low in individualism and masculinity. This means that the society fosters strong relationships where everyone takes responsibility for fellow members of their group and that dominant values are caring for others and the quality of life. These characteristics are coherent with the values of universalism and benevolence that we discover in the clip. Therefore, the video features a high degree of localization.

A final observation regards the effects of the values inscribed within informational resources. In the following we shall report a comment to the Thai clip made by a subject after having viewed the video on YouTube:

"This video touched me on an unthinkable level so much that I'm almost reevaluating my life right now on how I can do nicer things to people. I strongly believe they should

show this to as many people around the world no matter what religious beliefs or race or language they speak"

This is an example of consequential effect as described in Section III.

VI. DISCUSSION

We are actually experimenting the use of the ValueML language with a dataset of commercial videos collected from the web. The dataset includes videos advertising objects or services as well as videos that campaign for or against something, e.g., for preserving the environment or against bullying, and videos that communicate brand personality or identity. The aim is to validate the proposed markup language, to obtain insights about how to refine the language and to reach a better understanding of some critical aspects regarding the communication of values through video commercials.

The experiments show that specific pairs of valorizations very often come together. As an instance, a practical value is often accompanied by a critical one: the advertised good is not only valorized by emphasizing its utility and function but it is usually compared to other goods by presenting its characteristic features and benefits. Analogously, utopian valorizations are often presented with ludic ones. Although the current version of the ValueML allows the association of more than one value to the same segment of a clip, it seems to us that the simultaneous presence of these pairs introduces a precise semantic meaning that cannot be captured by the simple sum of the meanings of its component parts. By looking at recent semiotic literature, we found that Semprini proposed a classification of consumer's values that could be useful to address the above problem [37] [38]. The proposal, called "semiotic mapping of consumer's values", follows a dimensional approach and uses two pairs of values (borrowed from the Floch's inventory) to represent a bi-dimensional continuous space of possible valorizations. Four aggregated values emerge, that correspond to the four quadrants of the space. These are: information (practical and critical values); mission (critical and utopian values), project (utopian and ludic values), and euphoria (ludic and practical values). Semprini's model is interesting for its semantic interoperability with respect to the Floch's one. However, in order to exploit this feature the actual version of the ValueML language should be enriched by allowing the definition of dimensional models of values that are currently not yet supported. Another open issue regards the modality attribute. The experiments with narrative videos show that values can be expressed not only through the characters' behaviors as shown in the Thai clip but also by means of other narrative elements such as, for example, the kind of space and time represented in the story: the setting, the personality of the characters, the relations existing between the characters, the expressed passions, emotions, and events. Therefore, there is the need to further develop and articulate the set of possible modalities in order to take into account these possibilities.

An interesting open issue regards the links existing between values and emotions. According to appraisal theory [39], emotions are generated by the evaluation (an appraisal process) of a given situation, object, action (a stimulus) with respect to a personal value, goal, aesthetic or ludic preference, and expectations (a concern). This fact opens up the possibility of hypothesizing the value, given the emotion and the stimulus. This is important if we want to embed the ValueML in other languages such as the EmotionML language [8]. Are annotated emotions coherent with annotated values? Is it possible to infer the values from represented emotions (and vice-versa)?

As shown in Section V, the annotation of values can be exploited in cross-cultural design to select the resource that best match the culture of the destination audience. Moreover, the explicit representation of how a value has been inscribed in the information resource is valuable information for multimodal designers and a source of inspiration for the development of new resources [27]. Analogously, the annotation can be exploited to assess the consistency of the values communicated through a collection of advertisement products belonging to the same ad campaign or to study the diachronic evolution of valorizations of a brand in a given time interval. As an instance, it can be shown that Benetton started its ad campaigns in the 60s by exploiting practical and critical values then it moves towards ludic valorizations (in the 80s), and to utopian values in the 90s.

Finally, the presentation of the ValueML made in Section IV left a number of technical problems unsolved.

It is unclear, for example, if it is better to use continuous or discrete, unipolar or bipolar, scales for representing relevance and confidence. The current choice, a continuous unit-less scale such as [0,1], may be overly restricted. Human annotators often use a set of discrete labels such as a five-point Likert scale ("very low", "medium low", ... ," medium high", "very high"). Temporal intervals are currently represented using start and end time points that are measured with respect to the beginning of a clip. This is consistent with similar conventions used in other XML languages such as SMIL. Time instants are specified using the normal play code. Alternatives are SMPTE time code or real world clock time. Another problem regards the association of the same value to multiple time intervals. Should it be made using different value elements and media fragment URIs (one for each segment) or by a single value element and media fragment URI (see Figure 11)?

We have postponed a final decision about these topics after the acquisition of more knowledge.

VII. CONCLUSIONS

In this paper we address an aspect that has not been tackled before, namely, analysing and annotating web resources with embedded values. To this end, we propose a markup language, called ValueML, and illustrate its use in two paradigmatic cases. Some specifications are still sub optimal, such as the specification of multiple vocabularies; other aspects are currently missing but will be likely required by annotators such as the use of dimensional models of values or the use of qualitative scales. Nevertheless, experiments with video commercials have confirmed the feasibility of the approach and the effectiveness of this preliminary version of the language.

We highlight some aspects of the approach that we deem important. First, our interest is on model-based annotation made by professionals during resource production [27]. The term model-based is used, here, to stress the fact that analysis and annotation are driven by a metamodel of the genre of resource to be annotated (see Figure 3) and exploit theoretically founded value taxonomies. These tasks require specific competencies and a particular analytic sensibility since values are often embedded within implicit features of a resource (e.g., its atmosphere) that are not easily accessible by non expert people. The annotator may be the author/designer of the resource. In fact, she is in a privileged position to provide annotating knowledge. Other possibilities include multimedia critics, semioticians, commentators. If the interpretation is not the designer's one, then it should be, at least, an interpretation that the designer would consider as a possible one. The consequence is that for the same resource, several annotations could exist, each one representing a specific interpretation according to the interests, the perspective and the value taxonomy used by the annotator.

Second, manual annotation is time-consuming but the experiments with video commercials we have worked out showed that it is a feasible approach due to the limited time extension of these genres of media artifacts. The effort is largely rewarded by the benefits we may obtain that are numerous. In the field of Brand Marketing and Communication, for example, the use of ValueML allows producers to explicitly represent the brand identity core values that are inscribed within an advertising message. As a consequence, ad resources can be selected, compared, and aggregated to made portfolios on the base of embedded values. Moreover, value annotation makes is easier to evaluate the degree of consistency of ad messages communicated by different resources belonging to the same ad campaign or through different brand signals (e.g., product design, packaging, testimonials, flagship stores and exhibitions). In the field of Cross-Cultural Design, value annotation makes it easier to select the resource that best match a given context of use, i.e., a resource that could resonate with the values of the target consumers or with the values of a destination culture. In addition, the ValueML language allows the annotator to unfold design knowledge embedded within a resource. It allows, for example, to use the "`modality`" attribute in order to explain *how* a value is inscribed within the ad message. This is inspirational for designers who may exploit these insights for addressing new design problems. We claim that this kind of knowledge can be exploited also for automatic recognition of values. This is in line with a growing interest in automatic understanding of images and video advertisements as reported, for example, in [40]. Having a dataset of professionally annotated video commercials can be used, in the future, as a *ground truth* for supervised learning techniques. To this end, we claim that model based annotation could provide better results with respect to current approaches based on the exploitation of Amazon Mechanical Turk annotators.

Finally, the use of ValueML may help designers to better focalize their attention on the ethical aspects embodied in their products, and the final consumers to become more aware of the technological mediation effect produced by persuasive messages.

Our current work is devoted to enrich our dataset of annotated resources with new cases. We think that this phase is needed for stabilizing the language in a form suitable for final formalization. A further step will be the design and implementation of an ontology to define the terms of the ValueML language, to relate the terms to one another, and to define mappings between value vocabularies when possible. We are also investigating opportunities for promoting the standardization of the ValueML as a recommended format for value representation.

ACKNOWLEDGMENTS

The author would like to thank the editor and reviewers for their comments and suggestions. Special thanks also to Massimo Romanin for his help in developing the first version of the ValueML language.

REFERENCES

[1] M. Romanin and E. Toppano "Value annotation of web resources: the ValueML language," Proc. WEB 2018: The Sixth International Conference on Building and Exploring Web Based Environments, IARIA, Nice, France, pp. 32-37, 2018.

[2] J. Belman, H. Nissenbaum, M. Flanagan, and J. Diamond, "Grow-A-Game: a tool for values conscious design and analysis of digital games," Proc. of DiGRA Conference, Vol. 6, pp. 14-17, 2011.

[3] B. Friedman, P. H. Kahn, A. Borning, and A. Huldtgren, "Value sensitive design and information systems," in Early engagement and new technologies: opening up the laboratory, pp. 55-95, Springer, Netherlands, 2013.

[4] G. Cockton, "A development framework for value-centered design," Proc. CHI EA '05, pp. 1292-1295, 2005.

[5] P. M. A. Desmet and A. E. Pohlmeyer, "Positive Design: an introduction to Design for Subjective Well-Being," International Journal of Design, Vol. 7, No. 3, pp. 5-19, 2013.

[6] P. Brey, "Values in technology and disclosive computer ethics," The Cambridge Handbook of Information and Computer Ethics, Ed. L. Floridi, Cambridge University Press, pp. 41-58, 2009.

[7] F. Ceschin and I. Gaziulusoy, "Evolution of design for sustainability: from product design to design for system innovation and transition," Design Studies, 47, pp. 118-163, 2016.

[8] W3C. Emotion Markup Language (EmotionML), May 2014, URL: https://www.w3.org/TR/emotionml/ [retrieved: November, 2018].

[9] E. Cambria, "Affective computing and Sentiment analysis," IEEE Intelligent Systems, March/April 2016, pp. 102-107, 2016.

[10] S. Kujala and K. V.V.Mattila, "Value of Information Systems and Products: Understanding the Users' Perspective and Values," Journal of Information Technology Theory and Application, 9:4, pp. 23-39, 2009.

[11] S. Boztepe, "User Value: competing theories and models," International Journal of Design, Vol. 1, No.2, pp. 55-63, 2007.

[12] C. A. Le Dantec, E. S. Poole, and S. P. Wyche, "Values as lived experiences: evolving Value Sensitive Design in support of value discovery," Proc. CHI 2009, ACM, pp. 1141-1150, 2009.

[13] S. H. Schwartz, "An Overview of the Schwartz theory of Basic Values," Online Readings in Psychology and Culture, 2(1), Article 11, 2012, https://doi.org/10.9707/2307-0919.1116

[14] A. Cheng and K. R. Fleischmann, "Developing a meta-inventory of human values," ASIS&T, Vol. 47, No. 1, pp. 1-10, 2010.

[15] M. Rokeach. The nature of human values. New York: Free Press, 1973.

[16] P. Brey, "Theorizing the cultural quality of new media," Technè, 11:1, Fall 2007, pp. 2-18. URL: http://scholar.lib.vt.edu/ejournals/SPT/v11n1/brey.html [retrieved: November, 2018].

[17] M. Hollbrok. Consumer value: a framework for analysis and research, Routledge, New York, 1999.

[18] B. Friedman and P. H. Kahn, "Human values, Ethics, and Design, " in Handbook of Human-Computer Interaction, J. Jacko, A. Sears, and N.J. Mahwah, Editors, Lawrence Erlbaum, pp. 1177-1201, 2003.

[19] K. Sheldon, A. J. Elliot, Y. Kim, and T. Kasser, "What is satisfying about satisfying events? Testing 10 candidate psychological needs," Journal of Personality and Social Psychology, Vol. 80, No. 2, pp. 325-339, 2001.

[20] J. M. Floch, Semiotics, Marketing and Communication: beneath the signs, the strategies. Palgrave MacMillan, 2001.

[21] M. Flanagan, J. Belman, H. Nissenbaum, and J. Diamond, "A method for discovering values in digital games," Proc. DiGRA Conference, pp. 752-760, 2007.

[22] N. Crilly, A. Maier, and P.J. Clarkson, "Representing artifacts as media: modelling the relationship between designer intent and consumer experience," Journal of Design, Vol. 2, No. 3, pp. 15–27, 2008.

[23] M. Flanagan, D. C. Howe, and E. Nissenbaum, "Embodying values in technology. Theory and Practice," in: Information Technology and Moral Philosophy, (J. van de Hoven and J. Weckert Eds.), Cambridge: Cambridge University Press, pp. 322-353, 2008.

[24] L. Anticoli and E. Toppano, "Technological mediation of ontologies: the need for tools to help designers in materializing ethics," International Journal of Philosophy Study, Vol. 1, Issue 3, pp. 23-31, 2013.

[25] B. Fogg. Persuasive Technology: Using Computers to Change What We Think and Do. Morgan Kaufmann, San Francisco, 2003.

[26] R. H. Thaler and C. S. Sunstein. NUDGE. Improving decisions about health, wealth, and happiness, Yale University Press, 2008.

[27] E. Toppano and V. Roberto, "Semiotic annotation of narrative video commercials: bridging the gap between artifacts and ontologies," International Journal on Advances in Internet Technology, vol. 10, nr. 3&4, pp. 145-162, IARIA, 2017, http://www.iariajournals.org/internet_technology/ [retrieved: November, 2018].

[28] W3C. Media Fragments URI 1.0, September 2012, URL: https://www.w3.org/TR/media-frags/ [retrieved: November, 2018].

[29] C. Chang, "Narrative advertisements and narrative processing," in Advertising Theory (S. Rodgers and E. Thorson Eds.), Routledge, pp. 241-254, 2012.

[30] M. C. Green and T. C. Brock, "The role of transportation in the persuasiveness of public narratives," Journal of Personality and Social Psychology, Vol. 79, No. 5, pp. 701-721, 2000.

[31] C. Bianchi, "Semiotic approaches to advertising texts and strategies: narrative, passion, marketing," Semiotica 183, 1/4, pp. 243-271, 2011.

[32] Citroen Bx Campaign, YouTube, URL: https://www.youtube.com/watch?v=jH2HLRpIq2Y [retrieved: November, 2018].

[33] Thai Good Stories, YouTube, URL: https://www.youtube.com/watch?v=cZGghmwUcbQ [retrieved: November, 2018].

[34] S.H. Schwartz, G. Melech, A. Lehrnami, S. Burgess, M. harris, and V. Owens, " Extending the cross-cultural validity of the theory of basic human values with a different method of measurement," Journal of Cross-Cultural Psychology, 32, 519-542, 2001.

[35] B. Hellwing, and D. Uytvanck, "EUDICO Linguistic Annotator (ELAN)," Version 2.0.2, software manual, 2004.

[36] G. Hofstede, G.J. Hofstede, and M. Minkov, Cultures and Organizations. Software of the mind. Mc Graw Hill, 2010.

[37] A. Semprini, Marche e mondi possibili. Un approccio semiotico al marketing della marca. Franco Angeli, 2007.

[38] G. Rossolatos, "Applying structuralist semiotics to brand image research," The Public Journal of Semiotics, IV(1), pp. 25-82, 2012.

[39] E. Demir, P. M. A. Desmet, and P. Hekkert, "Appraisal patterns of emotions in human-product interaction," International Journal of Design, Vol. 3, No. 2, pp. 41-51, 2009.

[40] Z. Hussain, M. Zhang, X. Zhang, K. Ye, C. Thomas, Z. Agha, N. Ong, and A. Kovashka, "Automatic Understanding of Image and Video Advertisements, " Proc. IEEE 2017 Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1100-1110, 2017.

DXNet: Scalable Messaging for Multi-Threaded Java-Applications Processing Big Data in Clouds

Kevin Beineke, Stefan Nothaas and Michael Schöttner

Institut für Informatik, Heinrich-Heine-Universität Düsseldorf,
Universitätsstr. 1, 40225 Düsseldorf, Germany
E-Mail: [Kevin.Beineke,Stefan.Nothaas,Michael.Schoettner]@hhu.de

***Abstract*—Many big data and large-scale cloud applications are written in Java or are built using Java-based frameworks. Typically, application instances are running in a data center on many virtual machines which requires scalable and efficient network communication. In this paper, we present the practical experience of designing an extensible Java network subsystem, called DXNet, providing fast object de-/serialization, automatic connection management and zero-copy messaging. Additionally, we present a Java.nio-based transport for DXNet, called EthDXNet, to efficiently utilize Ethernet networks. The proposed design uses a zero-copy send and receive approach for asynchronous messages and requests/responses. DXNet is optimized for small messages (< 100 bytes) in order to support graph-based applications, but also works well with larger messages (e.g., 8 MB). DXNet is available on GitHub, and its modular design is open for different transport implementations currently supporting Ethernet and InfiniBand. EthDXNet is based on Java.nio socket channels complemented by application-level flow control to achieve low latency and high throughput for 10 GBit/s and faster Ethernet networks. Furthermore, a scalable automatic connection management and a low-overhead interest handling provide efficient network communication for dozens of servers, even for small messages (< 100 bytes) and an all-to-all communication pattern. The evaluation with micro-benchmarks and the Yahoo! Cloud Serving Benchmark (YCSB) shows the efficiency and scalability with up to 64 virtual machines in the Microsoft Azure cloud. Furthermore, DXNet achieves request-response latencies sub 10 µs (round trip) including object de-/serialization, as well as a maximum throughput of more than 9 GByte/s on a private cluster (using InfiniBand).**

Keywords–Message passing; Ethernet networks; InfiniBand; Java; Cloud computing.

I. INTRODUCTION

This paper is an extended version of the conference paper "Scalable Messaging for Java-based Cloud Applications" published at ICNS 2018 [1].

Big data processing is emerging in many application domains of which many are developed in Java or are based on Java frameworks [2][3][4]. Typically, these big data applications aggregate the resources of many virtual machines in cloud data centers (on demand). For data exchange and coordination of application instances, an efficient network transport is essential. Fortunately, public cloud data centers already provide 10 GBit/s Ethernet, 56 GBit/s InfiniBand and faster.

Java applications have different options for exchanging data between Java servers, ranging from high-level Remote Method Invocation (RMI) [5] to low-level byte streams using Java sockets [6] or the Message Passing Interface (MPI) [7]. However, none of the mentioned possibilities offer high performance messaging, elastic automatic connection management, advanced multi-threaded message handling and object serialization all together.

In this paper, we propose DXNet, a network messaging system which addresses all of these requirements. DXNet is a network library for Java-based applications which has originally been designed for DXRAM [8] a distributed in-memory key-value store and DXGraph [9] a graph processing framework built on top of DXRAM. We provide DXNet as a standalone library through GitHub [10] as we think it is useful for many other Java-based big data applications. DXNet is extensible by transport implementations to support different network interconnects. In this paper, we also present the Ethernet transport implementation for DXNet, called EthDXNet. The Ethernet transport is based on Java.nio and provides high throughput and low latency networking over Ethernet connections.

The contributions of this paper are:

- the DXNet architecture (highly concurrent and transport agnostic)
- zero-copy, parallel de-/serialization of Java objects
- lock-free, event-driven message handling
- scalable automatic connection management
- zero-copy approach for sending and receiving data over socket channels
- efficient socket channel interest handling
- evaluations with 5 GBit/s Ethernet (with up to 64 VMs in the Microsoft Azure cloud) and 56 GBit/s InfiniBand networks

The evaluation shows that DXNet efficiently handles high loads with dozens of application threads concurrently sending and receiving messages. Synchronous request/response patterns can be processed in sub 10 µs Round-Trip Time (RTT) with InfiniBand transport (including object de-/serialization). Also, high throughput is achieved even with smaller payloads, i.e., bandwidth saturation with 1-2 KB payload on InfiniBand and 256-byte payload on Ethernet. Furthermore, the evaluation shows that EthDXNet scales well while per-node message

throughput and request-response latency is constant from 2 to 64 nodes, even in an high-load all-to-all scenario (worst case).

The structure of the paper is as follows: after discussing related work, we present an overview of DXNet in Section III. In Section IV, we describe the lock-free Outgoing Ring Buffer followed by the concurrent serialization in Section V. The next section explains the event-driven processing of incoming data. Section VII presents thread parking strategies. In Section VIII, we describe the sending and receiving procedure of EthDXNet, followed by a presentation of the connection management in Section IX. Section X focuses on the flow control implementation and Section XI on the interest handling. Transport implementations for InfiniBand and Loopback are described in Section XII. Evaluation results are discussed in Section XIII, followed by the conclusion.

II. Related Work

In this section, we discuss related work for this paper. DXNet combines high-level thread and connection management and a concurrent object de-/serialization with lock-free, event-driven message handling and zero-copy data transfer over Ethernet and InfiniBand (extensible). To the best of our knowledge, no other Java-based network library provides this communication semantics. We compare DXNet with the most relevant related work, only.

A. DSM

Distributed Shared Memory (DSM) is re-gaining attraction due to fast networks supporting **RDMA** but is not an option for most existing Java applications because it requires many modifications within the Java Virtual Machine (JVM) and its memory management [11]. Furthermore, despite all advantages modern networks provide, DSM systems have limited scalability because of their transparent implicit communication [12].

B. Java RMI

Java's RMI [5] provides a high-level mechanism to transparently invoke methods of objects on a remote machine, similar to Remote Procedure Calls (RPC). Parameters are automatically de-/serialized, and references result in a serialization of the object itself and all reachable objects (transitive closure), which can be costly [13]. Missing classes can be loaded from remote servers during RMI calls which is very flexible but introduces even more complexity and overhead. The built-in serialization is known to be slow and not very space efficient [13][14]. Furthermore, method calls are always blocking.

Manta [15] improves runtime costs of RMI by using a native static compiler. **KaRMI** [16], a drop-in replacement for Java RMI, is implemented in Java without any native code supporting standard Ethernet. KaRMI also replaces Java's built-in serialization reducing overhead and improving overall performance. DXNet does not provide transparent remote method calls but an efficient parallel serialization which avoids copying memory. DXNet is primarily designed for parallel applications and high concurrency, RMI for Web applications and services.

C. MPI

MPI is the state-of-the-art message passing standard for parallel high-performance computing and provides very efficient message passing for primitive, derived, vector and indexed data types [17]. As MPI's official support is limited to C, C++ and Fortran, Java object serialization is not considered by the standard. Nevertheless, MPI is available for Java applications through implementations of the MPI standard in Java [18] or wrappers of a native library [19].

MPI-2 introduced multi-threading for MPI processes [17] enabling well-known advantages of threads. Prior to MPI-2, intra-node parallelization demanded the execution of multiple MPI processes (and the use of more expensive IPC). To enable multi-threading, the process has to call `MPI_init_thread` (instead of `MPI_init`) and to define the level of thread support ranging from single-threaded execution over funneled and serialized multi-threading to complete multi-threaded execution (every thread may call MPI methods at any time). A lot of effort has been put into the latter to provide a high concurrent performance [20][21]. Still, the performance is limited compared to a message passing service designed for multi-threading [20].

One of DXNet's main application domains are long running applications with dynamic node addition and removal (not limited to), e.g., distributed key-value stores or graph storages. The MPI standard defines the required functionality for adding and removing processes (over Berkeley Sockets with `MPI_Comm_join` or by calling `MPI_Open_port` and `MPI_Comm_accept` on the server and `MPI_Comm_connect` on the client). Unfortunately, most recent MPI implementations are still not fully supporting these features [22][23]. Furthermore, job shut down and crash handling is also limited [23]. MPI is particularly suitable for spawning jobs with finite runtime in a static environment. DXNet, on the other hand, was designed for up- and down-scaling and handling node failures. In [24], DXNet was used in the in-memory key-value store DXRAM to examine crash behavior and scalability.

D. Sockets

High level mechanisms for typical **socket-like interfaces** supporting Gigabit Ethernet (and higher) are provided by Java.nio [25][26], Java Fast Sockets (JFS) [27] or High Performance Java Sockets [28]. DXNet uses Java.nio to implement a transport for commonly used Ethernet networks.

1) Java.nio: The `java.io` and `java.net` libraries provide basic implementations for exchanging data via TCP/IP and UDP sockets over Input- and OutputStreams [25][6]. To create a TCP/IP connection between two servers, a new `Socket` is created and connection established to a remote IP and port. On the other end, a `ServerSocket` must be listening on given IP-port tuple creating a new socket when accepting an incoming connection-creation request. The connection creation must be acknowledged from both sides and can be used to exchange byte arrays by reading/writing from/to the socket hereafter. While this is sufficient for small applications with a few connections, this basic approach lacks several performance-critical optimizations [29] introduced with `Java.nio` [25][26]. (1) Instead of byte arrays, the read/write

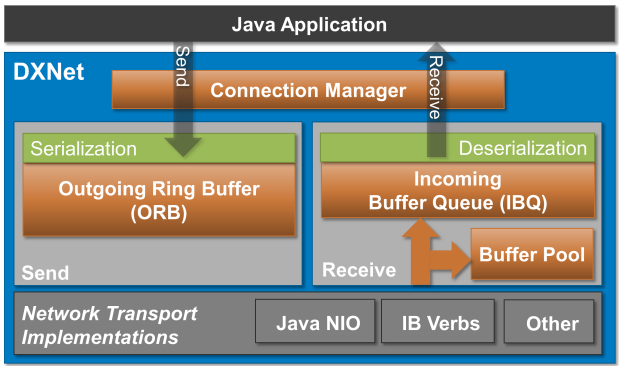

Figure 1. Simplified DXNet Architecture.

methods of Java.nio use `ByteBuffers`, which provide efficient conversion methods for all primitive data types. (2) ByteBuffers can be allocated outside of the Java heap allowing system-level I/O operations on the data without copying as the ByteBuffer is not subject to the garbage collection outside of the Java heap. This relieves the garbage collector as well as lowering the overhead with many buffers. (3) `SocketChannels` and `Selectors` enable asynchronous, non-blocking operations on stream-based sockets. With simple Java sockets, user-level threads have to poll (a blocking operation) in order to read data from a socket. Furthermore, when writing to a socket the thread blocks until the write operation is finished, even if the socket is not ready. With Java.nio, operation interests (like READ or WRITE) are registered on a selector which selects operations when they are ready to be executed. This enables efficient handling of many connections with a single thread. The dedicated thread is required to call the `select` method of the selector which is blocking if no socket channel is ready or returns with the number of executable operations. All available operations (e.g., sending/receiving data) can be executed by the dedicated thread, afterward.

2) Java Fast Sockets: JFS is an efficient Java communication middleware for high-performance clusters [27]. It provides the widely used socket API for a broad range of target applications and is compatible with standard Java compilers and VMs. JFS avoids primitive data type array serialization (JFS does not include a serializer), reduces buffering and unnecessary copies in the protocol and provides shared memory communication with an optimized transport protocol for Ethernet. DXNet provides a highly concurrent serialization for complex Java objects and primitive data types which avoids copying/buffering.

III. DXNET

DXNet relieves programmers from connection management, provides transferring Java objects (beyond plain Java.nio stream sockets) and allows the integration of different underlying network transports, currently supporting reliable verbs over InfiniBand and TCP/IP over Ethernet. In this section, we give a brief overview of the interfaces and functionality of DXNet (see Figure 1). Further implementation details can be found in the GitHub repository [10].

A. Basic Functionality

Automatic connection management. DXNet abstracts physical network addresses, e.g., IP/Port for Ethernet or GUID for InfiniBand, by using **node IDs**. The aforementioned node address mappings are registered in the library and are mutable for server up- and downscaling. A new connection is opened automatically when a message needs to be sent to another server which is not connected thus far. In case of errors, the library will throw exceptions to be handled by the application. Connections are closed based on a recently used strategy, if the configurable connection limit is exceeded, or in case of network errors which may be reported by the transport layer or detected using timeouts, e.g., absent responses.

Sending messages. DXNet sends messages asynchronously to one or multiple receivers but also provides blocking requests (to one receiver) which return when the corresponding response is received (DXNet transparently manages the association of responses and requests). **Messages are Java objects and serialized** by using DXNet's fast and concurrent serialization (providing default implementations for most commonly used objects, see Section V). The serialization writes directly into the Outgoing Ring Buffer (ORB) which aggregates messages for high throughput (see Section IV) and is allocated outside of the Java heap. Sending data is performed by a decoupled transport thread based on event signaling. DXNet also includes a flow control mechanism, which is described in Section X.

Receiving messages. When incoming data is detected by the network transport, it requests a pooled *native* memory buffer and copies the data into the buffer (see Section VI and Figure 1). By using a native memory buffer, we avoid burdening the Java garbage collector. The term *native* is described in Section IV-C. The buffer containing the received data is then pushed to the Incoming Buffer Queue (IBQ), a ring buffer storing references on buffers which are ready to be deserialized (see Section VI). The buffer pool and the IBQ are shared among all connections. The buffers of the IBQ are pulled and processed asynchronously by dedicated threads. Message processing includes parsing message headers, creating the message objects and deserializing the payload data. Finally, the **received message is passed back to the application (as a Java object)** using a pre-registered callback method.

A brief overview of DXNet's API is shown in Table I.

B. High Throughput and Low Latency

A key objective of DXNet is to provide high throughput and low latency messaging even for small messages found in many graph applications, for instance. We achieve this with a thread-based and event-driven architecture using lock-free synchronization, zero-copy, and zero-allocation.

Multithreading. All processing steps like serialization, deserialization, message transfer and processing are handled by multiple threads which are decoupled through events allowing high parallelism.

Lock-free event signaling. Dispatching processing events between threads is implemented using lock-free synchronization providing low-latency signaling. CPU load is managed without impairing latency by parking currently idling threads (described in Section VII).

TABLE I. DXNET'S APPLICATION INTERFACE

Method	Description
`new DXNet(config,nodeMap)`	initialize/configure (max. connections, server address mappings etc.)

`MyMessage extends Message/Request/Response`
`  exportObject(exporter)`
`  importObject(importer)`
`  sizeOfObject()`
	define message (serializable Java object) by implementing three methods
	serialize message with predefined methods from exporter
	deserialize message with predefined methods from importer
	return payload length

`sendMessage(message)`	send message asynchronously (receivers defined in message instance)
`sendSync(request,timeout)`	send request/response synchronously
`MyReceiver implements MessageReceiver` `  onIncomingMessage(message)`	receive messages/requests as Java objects pre-registered callback handler function

Fast serialization. DXNet implements fast serialization of complex data structures and writes data directly into an ORB. Many threads can access the ORB in parallel and ORBs are not shared between different connections increasing concurrency even more. The processing of incoming messages is also highly scalable because of the event-driven architecture.

Zero copy. DXNet does not copy data for messaging (except de-/serialization). For TCP/IP, we rely on Java's DirectByteBuffers and for InfiniBand on verbs pinning the buffers used by DXNet.

Zero allocation. DXNet uses object pooling wherever possible avoiding time-consuming instance creation and, even more important, not burdening the Java garbage collector which may block an application in case of low memory for up to multiple seconds.

C. Network Transport Interface

DXNet supports different underlying reliable network transports. The integration of a new transport protocol requires implementing just five methods:

- signal data availability on connection (callback)
- pull data from ORB and send it
- push received data to IBQ
- setup a connection
- close a connection

Sections VIII to XI present an transport for Ethernet networks.

IV. LOCK-FREE OUTGOING RING BUFFER

The Outgoing Ring Buffer (ORB) is a key component for outgoing messages and essential for providing high throughput and low latency. The latter is achieved by a highly concurrent approach based on lock-free synchronization.

Each connection has one dedicated ORB allowing concurrent processing of different connections. The ORB itself allows many application threads serializing their outgoing messages concurrently and directly into the ORB. The ORBs are allocated outside of the Java heap in native memory allowing zero-copy sending by the network transport. Directly serializing Java objects into the ORB is more efficient than serializing each object in a separate buffer and combining them later by copying these buffers. The ORB preserves message order as given by the application threads and aggregates implicitly multiple smaller messages in order to achieve high throughput. We decided to use lock-free synchronization for concurrency control which is more complex but highly efficient concerning latency compared to locks.

A. Basic Lock-Free Approach

The ORB has a configurable but fixed size and is accessed concurrently by several producers (application threads) and one consumer (dedicated transport thread for sending messages). The configurable buffer size limits the maximum number of messages/bytes to be aggregated. For our experiments (see Section XIII), we used 1 MB and 4 MB ORBs.

Figure 2 shows the ORB with three application threads producing data (serialization cores). All pointers move forward from left to right with a wraparound at the end. The white area between F_P and B_P is free memory.

Messages available for sending (fully serialized) are detected by the consumer (sending core) between B_P and F_C. The consumer sends aggregated messages and moves B_P forward accordingly but not beyond F_C. All messages between F_C and F_P are not yet ready for sending as parallel serialization is still in progress.

F_P is moved forward concurrently (if the buffer has enough space left) by the producers using a Compare-and-Set (CAS) operation, available in Java through the `Unsafe` class (see Section IV-C). Therewith, each producer can concurrently and safely store the position of F_P in a local variable F'_P and adjust F_P by the size of the message to be sent. All F'_P pointers (thread-local variables) are used by the associated producer for writing its serialization data concurrently at the correct position in the ORB. The light-colored arrows in Figure 2 show the starting point of each serialization core (producer) whereas the solid-colored ones show the current position. In the example, the purple producer finished its serialization first, and the green and orange producers are still serializing.

F_C is moved forward by producers when messages are fully serialized. In Figure 2, the purple producer finishes before the orange and green ones but cannot set F_C to F_P because the two preceding messages (from the other producers) have not been completely serialized yet. Each producer can easily detect unfinished preceding messages by comparing its starting point (light-colored arrow) with F_C. A naive solution lets fast producers wait for slower ones (e.g., with wait-notify semantics). As we do not want to impact latency, we cannot

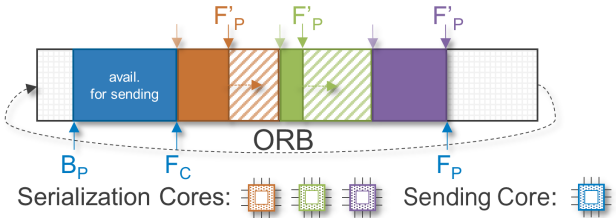

Figure 2. ORB for parallel serialization and aggregating outgoing messages. B_P: next message to consume. F_C: end of messages to consume. F_P: next free byte to produce. F'_P: thread-local copy F_P.

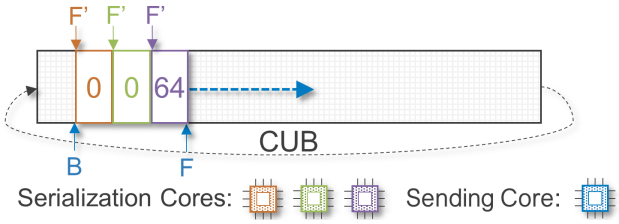

Figure 3. Catch-Up Buffer (CUB). Allowing faster producers returning early and not wasting CPU cycles for waiting. B: back pointer. F: front pointer. F': thread-local copy of F.

use locks/conditions here. An alternative solution is to busy-poll until all preceding messages have been serialized which stresses the CPU. A more sophisticated solution is presented in the next section.

B. Optimized Lock-Free Solution

The basic solution already avoids the overhead of locks, but with an increasing number of parallel serializations the probability of threads having to wait for slower ones increases (a thread might be slower because of less CPU time, for instance, or simply because the message to serialize is larger). The busy-polling can easily overload the CPU. Reducing the polling frequency of producers by sleeping (≥ 1 ms) or parking (≈ 10 µs) increases latency too much. Instead, we propose a solution which avoids having fast producers waiting for slower ones by leaving a *note* and returning early to the application. This note includes the message size so that slower producers can move F_P forward for the faster ones that already left. But, message ordering must be preserved.

Our solution is based on another configurable fixed-size ring buffer called Catch-Up Buffer (CUB). As mentioned before, we allocate one ORB for each connection which is now complemented by one associated CUB (e.g., with 1000 entries) for every ORB. The CUB is implemented using an integer array, each entry for one potential left-back note from faster producers. An entry will be 0 if there is no note or > 0 representing the message size if a producer finished faster than its predecessors. In the latter case, a slower producer will move forward F_P by the message size read from the CUB.

Figure 3 shows a CUB corresponding to the ORB shown in Figure 2. The front pointer F is moved concurrently using a CAS operation (similar to F_P in the ORB). The colored F' are the thread-local copies needed by the producers to leave back a potential note at the correct position in the CUB. The 64 is a note from the purple producer (message size of the filled purple box in Figure 2) who finished fastest and returned already to the application. The green and orange producers are still working (0 = no note). If the green producer would now finish before the orange one, it would also fill in its message size and return immediately.

If the orange producer finishes next, it moves forward F_C in the ORB as well as B in the CUB (leaving no note). The green one will do the same, but twice as it will detect the note (64) after committing its serialization and, thus, move forward F_C in the ORB by 64 bytes and also B by one slot (now pointing to F in the CUB, indicating we are done).

It is important that the order of entries in the ORB and the CUB is consistent, meaning, we need to move forward F and F_P, as well as B and F_C synchronously. We do this, by storing each of those two indexes in one 64-bit long variable in Java (e.g., F and F_P are stored together in one long) and, as the CAS operation works atomically on 64-bit longs, we can avoid locks.

Two more challenges remain, namely large messages which cannot be serialized at once and a potential ORB overflow during the serialization (both discussed in Section V).

C. Native Memory

The ORB, amongst other data structures, is allocated in native memory, i.e., it is located outside of the Java heap in the virtual address space of the Java process. Therefore, data structures stored in native memory are not managed by the JVM (e.g., no garbage collection or type safety). But, this does not pose a security risk for cloud applications as memory access is still managed by the operating system. On the other hand, using native memory allows the underlying network transports to send messages without copying them, for instance. The class `Unsafe` provides basic methods for memory allocation, memory copy and reading/writing primitives from/into native memory. Furthermore, `Unsafe` is very fast because of extensive optimizations and is widely used in third-party libraries [30].

We favor `Unsafe` over DirectByteBuffers [26] for two reasons. First, access is faster (e.g., missing boundary checks are already handled on a higher level). Second, `Unsafe` is more versatile because it allows accessing memory which was allocated in C/C++ code (e.g., used for InfiniBand).

V. Serialization

DXNet is designed to send and receive Java objects which need to be de-/serialized from/into a byte stream of messages. The built-in serialization of Java (interface `Serializable`) as well as file-based solutions are too slow and have a large memory footprint [31] (because of automatic un-/marshaling and the use of separators). Other binary serializer like Kyro [32], for instance, either do not support writing directly into native memory or interruptible processing which is needed by DXNet (see Sections V-A and V-B). We propose a new serializer addressing all these limitations while still being intuitive to use. The programmer has to implement two interfaces `Importable` and `Exportable`. The former requires implementing the method `importObject`, the latter `exportObject` and both `sizeOfObject`.

Figure 4. Message header.
Cat.: message, request or response; X: exclusive or not (ordering).

A. Export

Exporter. The serialization (or export) of Java objects requires an `exporter` which is passed to the `exportObject` method. The exporter class provides default method implementations for the serialization of all primitives, *compact numbers* and Strings and can be extended for supporting custom types (all types can also be arranged in arrays). Compact numbers are coded integers using a variable number of bytes as needed to reduce space overhead.

The exporter writes directly into the ORB by using Unsafe (see Section IV). It stores the start position within the ORB, the size of the ORB and the current position within the message.

Exporting an object involves two steps: exporting the message header (see Figure 4) which has a fixed size and exporting the variable-sized payload by calling the `exportObject` method.

DXNet uses its `default exporter` for serialization which is optimized for performance. It is complemented by two other exporters (described below) for handling messages which do not fit in the ORB without copying buffers.

Buffer overflow. If the end of the ORB will be reached during the serialization of an object, DXNet switches to the `overflow exporter`. The overflow exporter performs a boundary check for each data item of an object and writes bytes with a wrap-around to the beginning of the ORB, if necessary. The resulting message is sent as two pieces over the network stream avoiding copying data.

Large messages. Serialized objects resulting in messages larger than the ORB must be written iteratively. First, the entire unused section of the ORB (see Figure 2) is reserved and filled with the first part of the message. If the back pointer is reached, the export is interrupted and its current state is stored in an `unfinished operation` instance to allow resuming serialization as soon as there is free space in the ORB again.

Unfinished operation. The instance stores the interrupt position within the message and the rest of the current operation. Depending on the operation, the rest is either a part of a primitive which can be stored in a `long` within the unfinished operation or an object with partly uninitialized fields whose reference can be stored.

Resume serialization after an interrupt. To continue the serialization, the `exportObject` method is called again (threads return after being interrupted during serialization) and all previously successfully executed export operations are automatically skipped up to the position stored in the unfinished operation. The rest of the object is serialized from there (might be interrupted, again). For exporting large messages, the `large message exporter` is used, which extends the overflow exporter.

B. Import

All incoming messages are written into native memory buffers taken from the incoming buffer pool and are pushed to the IBQ (see Section VI). Each buffer contains received bytes (one or several messages) from the connection stream. The underlying network independently splits and aggregates packets resulting in a buffer beginning and ending at any byte within a message. DXNet is able to serialize split messages without copying buffers.

The import works analogously to the export. Messages are deserialized directly from native memory by using Unsafe (message header and payload). The fast `default importer` is complemented by three other importers (described below) for handling split messages. This requires to handle three situations: buffer overflow (tail of message/header missing), buffer underflow (head of a message/header is missing) and both combined.

Buffer overflow. When the buffer's end will be reached before the message is complete, we switch to the `overflow importer`. It does boundary checks and uses the unfinished operation (see Section V-A) when necessary. Furthermore, the serialization is aborted with an `IndexOutOfBoundsException` handled by DXNet avoiding returning invalid values for succeeding operations.

Buffer underflow. This situation occurs after a buffer overflow (on the same stream). It is known apriori and handled by the `underflow importer`, which uses the unfinished operation instance (passed from the overflow importer) containing all information necessary to continue deserialization.

Buffer under- and overflow. When a message's head and tail are missing (likely for large messages), the message is handled by the `underoverflow importer`.

C. Resumable Import and Export Methods

Messages may be split caused by DXNet's buffering or the underlying network. In order to avoid copying buffers, we require both import and export methods to be interruptible and idempotent as they may be called multiple times for one object (to avoid blocking threads, see Sections V-A and V-B). DXNet's importer and exporter methods are sufficient for most object types and only custom object structures must be aware of these requirements and avoid functions causing side effects (e.g., I/O access).

VI. Event-Driven Processing of Incoming Data

Figure 5 gives an overview of the parallel event-driven processing of incoming data. Like for the ORB, we use multithreading, lock-free synchronization, zero-copy and zero-allocation to provide high throughput and low latency.

Receiving process. The network transport pulls a buffer from the incoming buffer pool when new data can be received and fills it accordingly. The buffer is then pushed to the IBQ and processed by the **Message Creation Coordinator** thread (MCC) by deserializing the message headers. The message headers are pushed to the **message header store** afterward. Multiple message handler threads concurrently create the message objects, deserialize the messages' payloads

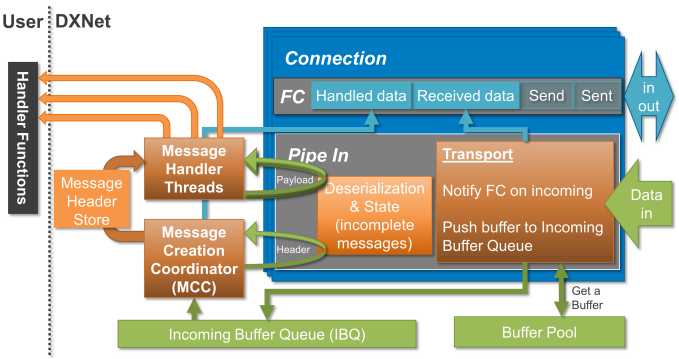

Figure 5. Receiving and processing messages. Green: Native memory access.

and pass the received Java objects to the application using its registered callback methods. When all data of a buffer has been processed, it is released and pushed back into the incoming buffer pool.

Incoming buffer pool. The buffer pool provides buffers, allocated in native memory, in different configurable sizes (e.g., 8×256 KB, 256×128 KB and 4096×16 KB). The transport pulls buffers using a worst-fit strategy as the number of bytes ready to be received on the stream is unknown. The pool can also scale-up dynamically when running out of buffers.

The buffer pool management consists of three lock-free ring buffers optimized for access of one consumer and N producers (similar to the ORB but without the CUB, see Section IV).

A. Parallel Message Deserialization

The transport thread pushes filled buffers into the IBQ. The IBQ is a basic ring buffer for one consumer and one producer and is synchronized using memory fences. The IBQ may be full and require the transport thread to park for a short moment and retry (see Section VII).

High throughput requires parallel deserialization. As the received messages of the incoming stream can be split over several incoming buffers (see Section V-B), the buffer processing must be in order and we need a two-staged approach to enable concurrency. The MCC thread pulls the buffer entries from the IBQ, deserializes all containing message headers (using relevant state information stored in the corresponding connection object) and pushes them into the message header store. Message payload deserialization based on the message headers can then be done in parallel by the message handler threads. This approach is efficient as the time-consuming payload deserialization and message object creation is parallelized.

The deserialization of split messages' payload (last message in the buffer, which is not complete) must be in order as well because all preceding parts of a message must be available to continue the deserialization of a split message. We address this situation by the MCC detecting and deserializing not only the header but the payload fraction within the current buffer, as well, for the split message. The rest of the message in the next buffer can be read by a message handler, again.

Split message headers are not an issue as deserialization of message headers is always done by the MCC which can store incomplete message headers within the connection object and continue with the next buffer.

Message header store. As mentioned before, the MCC pushes complete message headers to the message header store. The latter is implemented as a lock-free ring buffer for N consumers and one producer. Synchronization overhead is reduced by the MCC buffering the small message headers and pushing them in batches into the message header store. The batch size is limited but configurable, e.g., 25 headers.

Message header pool. Message headers are pooled, as well, in another single consumer, multiple producers lock-free ring buffer. Furthermore, message headers are pushed and pulled in batches. To reduce the probability of multiple message handler threads returning message headers at the same time, which increases latency because of collisions, the batch sizes differ slightly for every message handler.

Returning of buffers. A pooled buffer must not be returned before all its messages have been deserialized. Because of the concurrent deserialization and split messages, we use the MCC incrementing an atomic counter for every message header pushed to the message header store (more precisely, the counter is increased once for every batch of message headers). Accordingly, the message handlers decrement the counter for every deserialized message. When all messages have been deserialized, the buffer can be safely returned to the pool.

We could run out of buffers during high throughput if the MCC deserializes headers faster than the message handler threads can handle. Although we can scale up the number of incoming buffers, we prefer to throttle the MCC when a predefined number of used buffers is exceeded to reduce the memory consumption. Another benefit of limiting the number of incoming buffers is that all buffer states like the message counters, the buffers' addresses or the unfinished operations which are filled for incomplete messages can be allocated once and reused for every incoming buffer to be processed.

Message Ordering. DXNet allows applications to mark messages and thus ensure message ordering on a stream/connection. All marked messages are guaranteed to be processed (deserialized and executed) by the same message handler. All other steps preserve message ordering by default. For achieving maximum throughput, marking of messages should be used if necessary, only.

VII. Thread Parking Strategies

Lock-free programming allows low-latency synchronization but can easily overload a CPU by uncontrolled polling using CAS operations. DXNet implements a multi-level flow control with explicit message flow regulation and implicit throttling if memory pools drain and queues fill-up. We address three thread situations: blocked (the thread waits for another thread/server finishing its work because a pool is empty or queue full), colliding (failing CAS operation because another thread entered a critical section faster) and idling (the thread has nothing to do and waits for another thread/server committing new work).

Blocked thread. When blocked, the thread can park to reduce the CPU load because it was too fast executing its work compared to other threads/servers. However, the thread

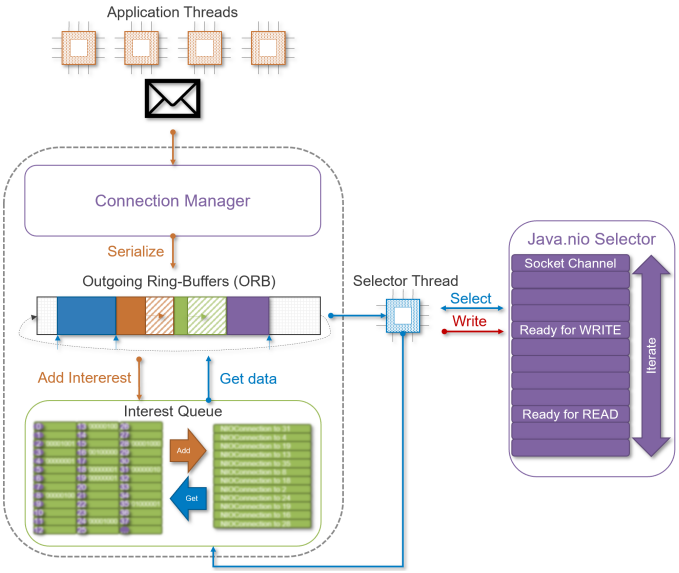

Figure 6. Data structures and Threads. Details of the Interest Queue can be found in Figure 8.

should not park for a long period to avoid restraining other threads/servers. Experiments showed that a reasonable park period is between 10 and 100 µs. Java allows minimum parking times of around 10 to 30 µs for a thread with `LockSupport.parkNanos()` for Linux servers with x86 CPUs.

Colliding thread. When colliding, the thread will repeat the CAS operation with updated values until successful because the thread is about to commit something and this should be done as fast as possible. However, reducing the collision probability (e.g., the ORB optimization described in Section IV-B) reliefs the CPU significantly.

Idling thread. This situation occurs, if a thread has nothing to do at the moment, e.g., a transport thread polls an empty ORB, the MCC polls an empty IBQ or a message handler polls an empty message header store. However, new work events can arrive within nanoseconds. Latency is minimized when threads do not park or yield, but only as long as the CPU is not overloaded. In the case of CPU overload situations, parking threads can reduce latency.

We address this with an overprovisioning detection combined with an adaptive parking approach (10 to 30 µs) if the number of active threads (application threads and network threads) reaches a threshold, e.g., four times the number of cores, see also Section XIII-A for the evaluation.

Idling for longer periods, e.g., applications not exchanging messages for a longer period, must be addressed, too. DXNet detects this, e.g., a network thread idling for one second (configurable time), and starts parking threads, if idling, reducing CPU load to a minimum.

VIII. EthDXNet - Sending and Receiving

In the following sections, we describe the Ethernet transport of DXNet, called EthDXNet. An overview of the most important data structures and threads of EthDXNet are depicted in Figure 6.

A. Sending of Data

To send messages, the DXNet API methods `sendMessage` or `sendSync` are called by the application threads (or message handler threads). In DXNet, messages are always sent asynchronously, i.e., application threads might return before the message is transferred. It is possible, though, to wait for a response before returning to the application (sendSync). After getting the `ConnectionObject` (a Java object) from the **Connection Manager**, the message is serialized into the ORB associated with the connection. For performance reasons, many application threads can serialize into the same or different ORBs in parallel (more in Section IV). The actual message transfer is executed by the **SelectorThread**, a dedicated daemon thread driving the Java.nio back-end. Thus, after serializing the message into the ORB, the application thread must signal data availability for the corresponding connection. This is done by registering a `WRITE` interest (see Table II) for given connection in the **Interest Queue** (see Section XI). When ready, Java.nio's **Selector** wakes-up the SelectorThread (which is blocked in the `select` method of the Selector) to execute the operation and thus to transfer the message.

After returning from the select method, a **SelectionKey** is available in the ready-set of the Selector. It contains the operation interest WRITE, the socket channel and attachment (the associated ConnectionObject). This SelectionKey is dispatched based on the operation. In order to send the message over the network, the SelectorThread pulls the **data block** from the ORB of the corresponding connection and calls the `write` method of the socket channel. From this point, we cannot distinguish single messages anymore because messages are naturally aggregated to data blocks in the ORBs, which is a performance critical aspect. The write method is repeatedly called until all bytes have been transferred or the method returned with return value 0. The second case indicates congestion on the network or the receiver and is best handled by stopping the transfer and continue it later. After sending, the back position (B_p, see Figure 2) of the ORB is moved by the number of bytes transferred to free space for new messages to send. Additionally, if the transfer was successful and the ORB is empty afterward, the SelectionKey's operation is set to `READ` which is the preset operation and enables receiving incoming data blocks. If the transfer failed, the connection is closed (see Section IX). If the transfer was incomplete or new data is available in the ORB, the SelectionKey is set to `READ | WRITE` (combination of READ and WRITE by using the bitwise or-operator) which triggers a new WRITE operation when calling select the next time but also allows receiving incoming messages. It is important to change the SelectionKey to this state as keeping only the WRITE operation could result in a deadlock situation in which both ends try to transfer data, but none of them can receive data on the same connection. This causes the **kernel socket receive buffers** to fill up on both sides preventing further data transfer.

The ORB is a ring buffer allocated in native memory (outside of the Java heap). In order to pass a ByteBuffer to the socket channel, which is required for data transfer, we wrap a **DirectByteBuffer** onto the ORB and set the ByteBuffer's position to the front position in the ORB and the limit to the back position. A DirectByteBuffer

is a ByteBuffer whose underlying byte array is stored in native memory and is not subject to garbage collection. This enables native operations of the operating system without copying the data first. The socket channel's send and receive operations are examples for those native operations, thus, benefiting from the DirectByteBuffer. Java does not support dynamically changing the address of a ByteBuffer. Therefore, on initialization of the ORB, we allocate a new DirectByteBuffer by calling `allocateDirect` of the Java object ByteBuffer and use the underlying byte array as the ORB. To do so, we need to determine the memory address of the byte array, which can be obtained with `Buffer.class.getDeclaredField("address")`. That is, during serialization the ORB is accessed with `Java.unsafe` by reading/writing from/to the actual address outside of the Java heap, but the socket channel accesses the data by using the DirectByteBuffer's reference (with adjusted position and limit). We do not access the ORB by using the DirectByteBuffer during serialization because of performance and compatibility reasons.

Although this approach prevents copying the data to be sent on user-level, the data is still copied from the ORB to the **kernel socket send buffer** which is a necessity of the stream-based socket approach. Therefore, correctly configuring the kernel socket buffer sizes (one for sending and one for receiving) has a significant impact on performance. We empirically determined setting both buffer sizes to the ORBs' size offers good performance without increasing the memory consumption too much (typically the ORBs are between 1 and 4 MB depending on the application use case).

B. Receiving of Data

Receiving messages is always initiated by Java.nio's **Selector** which detects incoming data availability on socket channels. When a socket channel is ready to be read from, the SelectorThread selects the SelectionKey and dispatches the READ operation. Next, the SelectorThread reads repeatedly by calling the `read` operation on the socket channel until there is nothing more to read or the buffer is full. If reading from the socket channel failed, the socket channel is closed. Otherwise, the ByteBuffer with the received data is flipped (limit = position, position = 0) and pushed to the IBQ (see Figure 1). The buffer processing is explained in Section VI.

In order to read from a socket channel, a ByteBuffer is required to write the incoming data into. Continually allocating new ByteBuffers would decrease the performance drastically. Therefore, we implemented a **buffer pool**. The buffer pool provides DirectByteBuffers in different configurable sizes (e.g., 8×256 KB, 256×128 KB and 4096×16 KB). The SelectorThread pulls them using a worst-fit strategy as the number of bytes ready to be received on the stream is unknown. It can also scale-up dynamically, if necessary. The buffer pool management consists of three lock-free ring buffers optimized for access of one consumer and N producers, see Section VI.

The pooled DirectByteBuffers are wrapped to provide the ByteBuffer's reference as well as the ByteBuffer's address. The reference is used for reading from the socket channel and the address is necessary to deserialize the messages from the ByteBuffer.

IX. Automatic Connection Management

For sending and receiving messages, DXNet has to manage all open connections and create/close connections on demand. A connection is represented by an object (ConnectionObject), containing a node ID to identify the connection based on the destination, a **PipeIn** and a **PipeOut**. The PipeOut consists mostly of an ORB, a socket channel and flow control for outgoing data. The PipeIn contains a socket channel, flow control for incoming data, has access to the buffer pool (shared among all connections) and more data structures important to buffer processing, which are not further discussed in this paper.

1) Connection Establishment: Connections are created in two ways: (1) actively by creating a new connection to a remote node or (2) passively by accepting a remote node's connection request. In both cases, the connection manager must be updated to administrate the new connection. Figure 7 shows the procedure of creating a new connection (active on the left side and passive on the right). The core part is the TCP handshake, which can be seen in the middle.

Active connection creation: A connection is created actively if an application thread wants to send a message to a not yet connected node. To establish the connection, the application thread creates a new ConnectionObject (including PipeIn and PipeOut and all its components), opens a new socket channel and connects the socket channel to the remote node's IP and port. Afterwards, the application thread registers a `CONNECT` operation, creates a `ReentrantLock` and `Condition` and waits until the Condition is signaled or the connection creation was aborted (using a lock here is not a performance issue as connections are held open). To correctly identify the corresponding ConnectionObject to a socket channel, the ConnectionObject is attached to the SelectionKey when registering the CONNECT interest and all following interests.

The SelectorThread continues the connection establishment by applying the CONNECT interest and selecting the socket channel when the remote node accepted the connection or the connection establishment failed. After selecting the SelectionKey, the socket channel's status is checked. If it is pending, the connection creation was successful so far, and the socket channel can be completed by calling `finishConnect`. If the connection establishment was aborted, the application thread is informed by setting a flag (which is checked periodically by the application thread).

The remote node has to identify the new node currently creating a connection. Thus, the node ID is sent to the remote node using the newly created channel. Furthermore, the SelectorThread marks the PipeOut as connected and signals the condition so the application thread can continue. It adds the connection to the connection manager, increments the connection counter and starts sending data, afterward.

Passive connection creation: For accepting and creating an incoming connection, the Selector implicitly selects a SelectionKey with `ACCEPT` operation interest which is processed by the SelectorThread by calling `accept` on the socket channel. This creates a new socket channel and acknowledges the connection. Afterward, the interest `READ` is registered in order to receive the node ID of the remote node. After selecting and

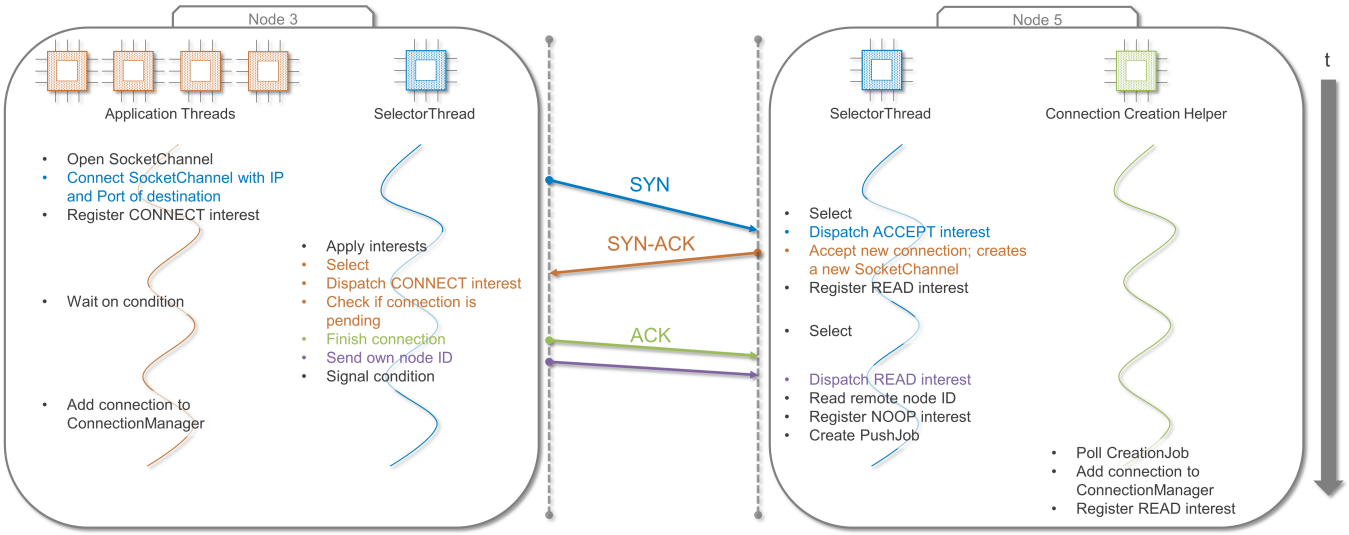

Figure 7. Connection Creation

dispatching the interest, the node ID is read by using the socket channel's read method.

At this point, the socket channel is ready for sending and receiving data, but the connection object has yet to be created and pushed to the connection manager. This process is rather time-consuming and might be blocked if an application thread creates a connection to the same node at the same time (connection duplication is discussed in Section IX-2). Therefore, the SelectorThread creates a job for creating the connection and forwards it to the **ConnectionCreationHelper** thread. Additionally, the interest is set to `NO-OP` (0) to avoid receiving data before the connection setup is finished and the connection is attached to the SelectionKey.

The ConnectionCreationHelper polls the job queue periodically. There are two types of jobs: (1) a connection creation job and (2) a connection shutdown job. The latter is explained in Section IX-3. When pulling a connection creation job, the ConnectionCreationHelper creates a new ConnectionObject (including the pipes, ORB, FC, etc.) and registers a READ interest with the new ConnectionObject attached. Furthermore, the PipeIn is marked as connected.

To be able to accept incoming connection requests, every node must open a `ServerSocketChannel`, bind it to a well-known port and register the `ACCEPT` interest. Furthermore, for selecting socket channels, a `Selector` has to be created and opened.

2) Connection Duplication: It is crucial to avoid connection duplication which occurs if two nodes create a connection to each other simultaneously. In this case, the nodes might use different connections to send and receive data which corrupts the message ordering and flow control. There are two approaches for resolving this problem: (1) detecting connection duplication during/after the connection establishment and (2) avoiding connection duplication by using two separate socket channels for sending and receiving.

Solution 1: **Detect and resolve connection duplication** by keeping one connection open and closing the other one. Apparently, the other node must decide consistently which can be done by considering the node IDs (e.g., always keep the connection created by the node with higher node ID). One downside of this approach is the complex connection shutdown. It must ensure that all data appended to the ORB of the closing connection, to be sent over the closing connection, has already been sent and received. Furthermore, message ordering cannot be guaranteed until the connection duplication situation is resolved.

Solution 2: **Avoid connection duplication** by using two socket channels per connection: one for sending and one for receiving (implemented in EthDXNet). Thus, simultaneous connection creation leads to **one** ConnectionObject with opened PipeIn and PipeOut (one socket channel, each) whereas a single connection creation opens either the PipeOut (active) or PipeIn (passive). This approach requires additional memory for the second socket channel, Java.nio's Selector has more socket channels to manage and connection setup is required from both ends. The additional memory required for the second socket channel is negligible as the kernel socket buffers are configured to use a little socket receive buffer for the outgoing socket channel and a little socket send buffer for the incoming socket channel. The second TCP handshake (for connection creation, both sides need to open and connect a socket channel) is also not a problem as both socket channels can be created simultaneously and for a long-running big data application connections among application instances are typically kept over the entire runtime. Finally, the overhead for Java.nio's Selector is difficult to measure but is certainly not the bottleneck taking into account the limitations of the underlying network latency and throughput. **Sending out-of-band (OOB) data** is possible by utilizing **the unused back-channel of every socket channel**. We use this for sending flow control data in EthDXNet (see Section X).

3) Connection Shutdown: Connections are closed on three occasions: (1) if a write or read access to a socket channel failed, (2) if a new connection is to be created, but the configurable connection limit is reached or (3) on node shutdown. In the first case, the SelectorThread directly shuts down the connection. In the second case, the application thread registers a `CLOSE` interest to let the SelectorThread close the connection

asynchronously. On application shutdown, all connections are closed by one **Shutdown Hook** thread.

To shut down a connection, first, the outgoing and incoming socket channels are removed from the Selector by canceling the SelectionKeys representing a socket channel's registration. Then, the socket channels are closed by calling the socket channels' `close` method. At last, the connection is removed from the connection manager by creating a shutdown job handled by the ConnectionCreationHelper (case (1)) or directly removing it when returning to the connection management (cases (2) and (3)). The ConnectionCreationHelper also triggers a `ConnectionLostEvent`, which is dispatched to the application for further handling (e.g., node recovery).

When dismissing a connection (case (2)), directly shutting down a connection might lead to data loss. Therefore, the connection is closed gracefully by waiting for all outstanding data (in the connection's ORB) to be sent. Priorly, the connection is removed from connection management to prevent further filling of the ORB. Afterward, a CLOSE interest is registered to close the socket channels asynchronously. The SelectorThread does not shut down the socket channels on the first opportunity but postpones shutdown for at least two RTT timeouts to ensure all responses are received for still outstanding requests.

X. Flow Control

DXNet implements a **flow control (FC)** protocol to avoid flooding a remote node with messages. This would result in an increased overall latency and lower throughput if the remote node cannot keep up with processing incoming messages. When sending messages, the per-connection dedicated FC checks if a configurable threshold is exceeded. This threshold describes the **number of bytes sent by the current node but not fully processed by the receiving node**. The receiving node counts the number of bytes received and sends a confirmation back to the source node in regular intervals. Once the sender receives this confirmation, the number of bytes sent but not processed is reduced. If an application send thread was previously blocked due to exceeding this threshold, it can now continue with processing the message.

EthDXNet uses the *Transmission Control Protocol* (TCP) which already implements a flow control mechanism on protocol layer. Still, DXNet's flow control is beneficial when using TCP. If the application on the receiver cannot read and process the data fast enough, the sender's TCP flow control window, the maximum amount of data to be sent before data receipt has to be acknowledged by the receiver, is reduced. The decision is based on the utilization of the corresponding kernel socket receive buffer. In DXNet, reading incoming data from kernel socket receive buffers is decoupled from processing the included messages, i.e., many incoming buffers could be stored in the IBQ to be processed by another thread. Thus, the kernel socket receive buffers' utilizations do not necessarily indicate the load on the receiver leading to delayed or imprecise decisions by TCP's flow control.

This section focuses on the implementation of the flow control in EthDXNet. Flow control data has to be sent with high priority to avoid unintentional slow-downs and fluctuations regarding throughput and latency. Sending flow control data in-band, i.e., with a special message appended to the data stream, is not an option because the delay would be too high. TCP offers the possibility to send **urgent data**, which is a single byte inlined in the data stream and sent as soon as possible. Furthermore, urgent data is always sent, even if the kernel socket receive buffer on the receiver is full. To distinguish urgent data from the current stream (urgent data can be at any position within a message as the transfer is not message-aligned), a dedicated flag within the TCP header needs to be checked. This flag indicates if the first byte of the packet is urgent data. Unfortunately, Java.nio does not provide methods for handling incoming TCP urgent data.

We solve this problem **by using both unused back-channels** of every socket channel which are available because of the double-channel connection approach in EthDXNet. Thus, the incoming stream of the outgoing socket channel and the outgoing stream of the incoming socket channel of every connection are used **for sending/receiving flow control data**.

Sending flow control data: When receiving messages, a counter is incremented by the number of received bytes for every incoming buffer. If the counter exceeds a configurable threshold (e.g., 60% of the flow control window), a `WRITE_FC` interest is registered. This interest is applied, selected and dispatched like any other WRITE interest. But, instead of using the socket channel of the PipeOut, **the PipeIn is used to send the flow control data**. The flow control data consists of one byte containing the number of reached thresholds (typically 1). If the threshold is smaller than 50%, for example, 30%, it is possible that between registering the WRITE_FC interest and sending the flow control data, the threshold has been exceeded again. For example, if the current counter is at 70% of the windows size which is more than two thresholds of 30%. In this case 2 * 30% = 60% is confirmed by sending the value 2. After sending flow control data, the SelectionKey is reset to READ to enable receiving messages on this socket channel, again.

Receiving flow control data: To be able to receive flow control data, the socket channel of **the PipeOut must be readable** (register READ). If flow control data is available to be received, the socket channel is selected by the Selector, and the SelectorThread reads the single byte from the socket channel of the PipeOut. When processing serialized messages on the sender, a counter is incremented. Application threads which want to send further messages if the counter reached the limit (i.e., the flow control window is full) are blocked until the receiver acknowledges message receipt. The read flow control value is used to decrement the counter to re-enable sending messages. Usually, the limit is never reached as the flow control data is received before (if the threshold on the receiver is low enough).

In Section VIII-A, we discussed the end-to-end situation of both nodes sending data to each other, but never reading (if the SelectionKey's operation stays at WRITE) causing a deadlock. This situation cannot occur with two socket channels per connection as reading and writing are handled independently. But, a similar situation is possible where two nodes send data to each other, but flow control data is not read for a while. This does not cause a deadlock but decreases performance. By setting the interest to READ | WRITE, flow control data is read from time to time ensuring contiguous high throughput.

TABLE II. JAVA.NIO INTERESTS

Interest	Description
OP_READ	channel is ready to read incoming data
OP_WRITE	set if data is available to be sent
OP_CONNECT	set to open connection
OP_ACCEPT	a connect request arrived
NO-OP	do nothing

TABLE III. ETHDXNET INTERESTS

Interest	Description (refers to attached connection)
CONNECT	set OP_CONNECT for outgoing channel
READ_FC	set OP_READ for outgoing channel
READ	set OP_READ for incoming channel
WRITE_FC	set OP_WRITE for incoming channel
WRITE	set OP_WRITE for outgoing channel
CLOSE	shutdown both socket channels

XI. EFFICIENT MANAGEMENT OF OPERATION INTERESTS

Operation interests are an important concept in Java.nio and are registered in the Selector to create and accept a new connection, to write data or to enable receiving data. The operation interests are complemented by the ConnectionObject (as an attachment) and the socket channel stored together in a SelectionKey. As soon as the socket channel is ready for any registered operation, the Selector adds the corresponding SelectionKey to a ready-set and wakes-up the SelectorThread waiting in the `select` method. If the SelectorThread is not waiting in the select method, the next select call will return immediately. The SelectorThread can then process all SelectionKeys.

A. Types of Operations Interests

The operation interests can be classified into two categories: **explicit operation interests and implicit operation interests**. Implicit operations are registered as presets after socket channel creation and after executing explicit operations. For example, a READ interest is registered for a socket channel if data is expected to arrive on this socket channel. The operation is then selected implicitly by the Selector whenever data is available to be received. Another example is the ServerSocketChannel which implicitly accepts new incoming connection requests if the ACCEPT interest has been registered before. Explicit operations are single operations which need to be triggered explicitly by the application. For example, when the application wants to send a message, the application thread registers a WRITE interest. When the socket channel is ready, the data is sent and the socket channel is set to the preset (in our case READ). It is not forbidden by Java.nio to keep explicit operations registered. But, as a consequence, the operations are always selected (every time select is called) which increases CPU load and latency. Therefore, in EthDXNet, every explicit operation is finished by registering an implicit operation.

The set of Java.nio operation interests is extended by EthDXNet to support flow control and to enable closing connections asynchronously. Table II shows all interests specified by Java.nio and Table III lists all interests used in EthDXNet. The interests READ, WRITE and CONNECT are directly mapped onto OP_READ, OP_WRITE and OP_CONNECT. OP_ACCEPT is registered and selected by the Selector and must not be registered explicitly. READ_FC and WRITE_FC are used to register OP_READ and OP_WRITE interests for the back-channel used by the flow control. The interest CLOSE does not have a counterpart because the method `close` can be called explicitly on the socket channel.

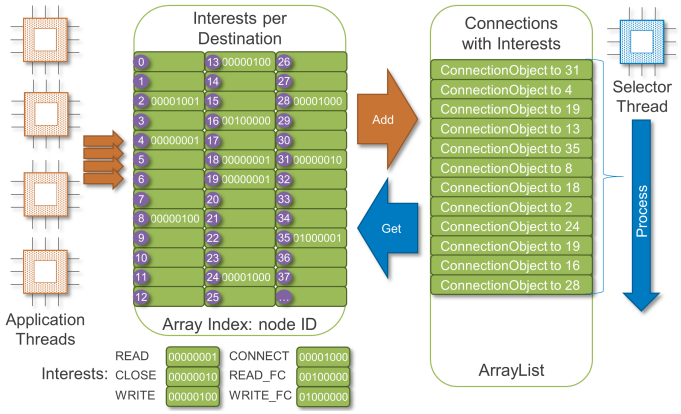

Figure 8. Interest Queue: the application threads add new interests to the Interest Queue. If interest was 0 before, the ConnectionObject is added to an ArrayList.

B. Interest Queue

None of the interests in Table III are registered directly to the Selector because only the SelectorThread is allowed to add and modify SelectionKeys. This is enforced by the Java.nio implementation which blocks all register calls when the SelectorThread is waiting in the `select` method. This obstructs the typical asynchronous application flow and can even result in a deadlock if the Selector does not have implicit operations to select. This problem can be avoided by always waking-up the SelectorThread before registering the operation interest and synchronizing the register and select calls. However, this workaround results in rather high overhead and a complicated workflow. Instead, **we address this problem with an Interest Queue** (see Figure 8) and register all interests in one bulk operation executed by the SelectorThread before calling select. This approach provides several benefits while solving the above problem: first, **the application threads can return quickly** after putting the operation interest into the queue and even faster (without any locking) when the interest was already registered (which is likely under high load). Second, **the operation interests can be combined** and put in a semantic order (e.g., CONNECT before WRITE) before registering (a rather expensive method call). Finally, **the operation interest-set can be easily extended**, e.g., by a CLOSE operation interest to asynchronously shut down socket channels.

Figure 8 shows the Interest Queue consisting of a byte array storing the operation interests of all connections (left side in Figure 8) and an ArrayList of ConnectionObjects containing connections with new operation interests sorted by time of occurrence (right side in Figure 8).

The byte array has one entry per node ID providing access

time in O(1). In DXNet, the node ID range is limited to 2^{16} (allowing max. 65,536 nodes per application). Thus, the operation interests of all connections can be stored in a 64-KB byte array. An array entry is not zero if at least one operation interest was added for given connection to the associated node ID. Operation interests are combined with the bitwise or-operator to avoid overwriting any interest. By combining operation interests, the ordering of the interests for a single connection is lost. But, this is not a problem because a semantic ordering can be applied later when processing the interests.

The ordering within the interests of one connection can be reconstructed but not the ordering across different connections. Therefore, whenever an interest is added to a non-zero entry of the byte array, the corresponding ConnectionObject is appended to an ArrayList. The order of operation interests is then ensured by processing the interest entries in the ArrayList in ascending order. The ArrayList also allows the SelectorThread to iterate only relevant entries and not all 2^{16}.

Processing operation interests: The processing is initiated either by the Selector implicitly waking up the SelectorThread if data is available to be read or an application thread explicitly waking up the SelectorThread if data is available to be sent. As waking-up the SelectorThread is a rather expensive operation (a synchronized native method call), it is essential to call it if necessary, only. Therefore, the SelectorThread is woken-up after adding the first operation interest to the Interest Queue across all connections (the ArrayList is empty after processing the operation interests), only. If the SelectorThread is currently blocked in the select call, it returns immediately and can process the pending operation interests.

Figure 9 shows the basic processing flow of the SelectorThread. The first step in every iteration is to register all operation interests collected in the ArrayList of the Interest Queue. The SelectorThread gets the destination node ID from the ConnectionObject and the interests from the byte array. Operation interests are registered to the Selector in the following order:

1) CONNECT: register SelectionKey OP_CONNECT with given connection attached to an outgoing channel.
2) READ_FC: register SelectionKey OP_READ with given connection attached to an outgoing channel.
3) READ: register SelectionKey OP_READ with given connection attached to an incoming channel.
4) WRITE_FC: change SelectionKey of an incoming channel to OP_WRITE if it is not OP_READ | OP_WRITE.
5) WRITE: change SelectionKey of an outgoing channel to OP_WRITE if it is not OP_READ | OP_WRITE.
6) CLOSE: keep interest in queue for delay or close connection (see Section IX-3).

The order is based on following rules: (1) a connection must be established before sending/receiving data, (2) setting the preset READ is done after connection creation, only, (3) all READ and WRITE accesses must be finished before shutting down the connection and (4) the flow control operations have a higher priority than normal READ and WRITE operations. Furthermore, re-opening a connection cannot be done before the connection is closed and closing a connection is only possible if the connection has been established before. Therefore it is not possible to register CONNECT and CLOSE together.

```
1  while (!closed) {
2   processInterests();
3
4   if (Selector.select() > 0) {
5    for (SelectionKey key :
         Selector.selectedKeys()) {
6     // Dispatch key
7     if (key.isValid()) {
8      if (key.isAcceptable()) {
9       accept();
10     } else if (key.isConnectable()) {
11      connect();
12     } else if (key.isReadable()) {
13      read();
14     } else if (key.isWritable()) {
15      write();
16     }
17    }
18   }
19  }
20 }
```

Figure 9. Workflow of SelectorThread.

Finally, the processing of registered operation interests includes resetting the operation interest in the byte array and removing the ConnectionObject from the ArrayList.

XII. Other Transport Implementations

DXNet has an open architecture supporting different network transport technologies. Currently, we have transport implementations for TCP/IP over Ethernet (described in Section VIII), reliable verbs over Infiniband (using native verbs over JNI), and Loopback (baseline for evaluation). We only sketch some important aspects of the InfiniBand transport here, as it will be published separately.

The `InfiniBand` transport accesses the IBDXNet library (C++) using JNI. IBDXNet utilizes ibverbs to implement direct communication using the InfiniBand HCA. IBDXNet uses one dedicated send and one dedicated receive thread, both processing outgoing/incoming data in native memory. Context switching from C++ to Java was designed carefully and is highly optimized to avoid latency.

The `Loopback` transport is used for the experiments in this paper allowing to study the performance of DXNet without any bottlenecks from a real network. Data is not sent over a network device nor the operating system's loopback device (latency would be considerably high) but is directly copied from the ORB to a pooled incoming buffer. Furthermore, the Loopback transport simulates a server sending and receiving messages at highest possible throughput allowing to evaluate DXNet's performance.

XIII. Evaluation

We evaluate the proposed concepts using DXNet's Loopback transport and three different networks: 1 GBit/s Ethernet, 5 GBit/s Ethernet (a shared 10 GBit/s Ethernet connection) and 56 GBit/s InfiniBand. The Loopback is used to evaluate

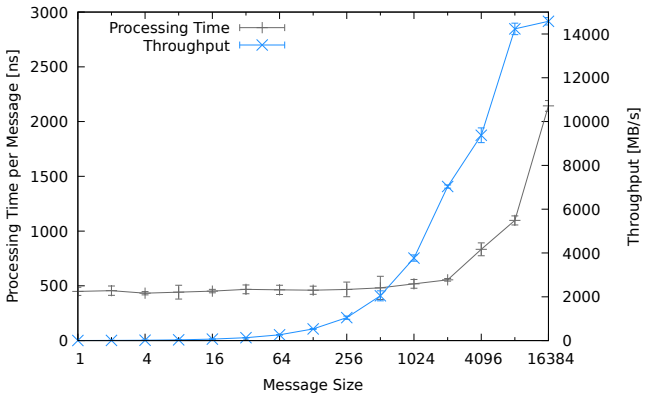

Figure 10. 10^7 Messages, 1 App. Thread, 4 Message Handlers.

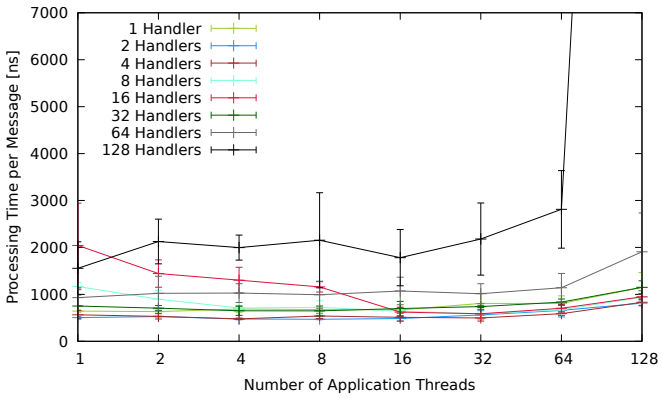

Figure 11. 10^7 Messages, 1024 Bytes Payload.

DXNet's concepts without any limitations of an underlying network. The Ethernet networks are used to evaluate EthDXNet.

Loopback and 5 GBit/s Ethernet tests were run in Microsoft's Azure cloud in Germany Central with up to 65 virtual machines (64 running the benchmark and one for deployment) from the type Standard_DS13_v2 which are memory optimized servers with 8 cores (Intel Xeon E5-2673), 56 GB RAM and shared 10 GBit/s Ethernet connectivity (two instances per connect). We deployed a custom Ubuntu 14.04 image with a 4.4.0-59 kernel and Java 8. In order to manage the servers, we created two identical scale-sets (as one scale-set is limited to 40 VMs). The tests with 1 GBit/s Ethernet and InfiniBand were executed on our private cluster servers with 64 GB RAM, Intel Xeon E5-1650 CPU and Ubuntu 16.04 with kernel 4.4.0-64.

We use a set of micro-benchmarks for the evaluation sending messages or requests of variable size with a configurable number of application threads. All throughput measurements refer to the payload size which is considerably smaller than the full message size, e.g., a 64-byte payload results in 115 bytes to be sent on IP layer when using Ethernet. Additionally, all runs with DXNet's benchmarks are **full-duplex** showing the aggregated performance for concurrently sending and receiving messages/requests.

A. *Loopback Transport*

As mentioned before, we want to evaluate the efficiency of DXNet's concepts without any network limitations. Figure 10 shows message processing times and throughputs for different

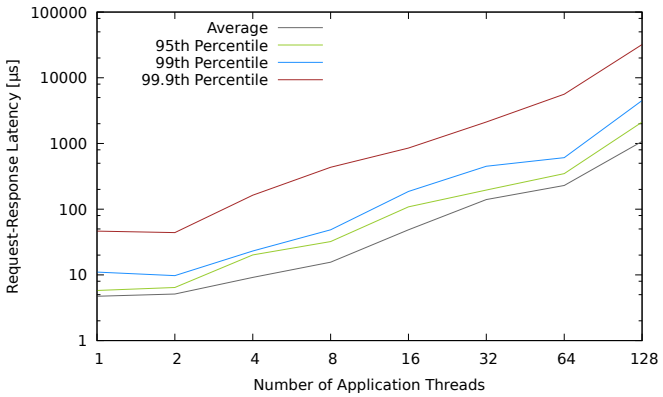

Figure 12. 10^6 Requests, 2 Message Handlers, 1 Byte Payload.

message sizes when using the Loopback transport on a typical cloud server (Standard_DS13_v2). Messages up to 2 KB can be processed in around 500 ns. Larger messages require increasing processing times, as expected. The throughput increases linearly with the message size up to 8 KB messages and is capped at around 14 GByte/s aggregated throughput for sending and receiving of larger messages. The Linux tool `mbw` determined a memory bandwidth of 7.19 GByte/s for a 16 GB array and 16 KB block for the used servers which explains the maximum throughput (saturation of the available memory bandwidth).

In Figure 10, we studied messages with up to 16 KB payload size as DXNet is primarily designed to perform well with small messages. We also tested larger messages (larger than the ORB, configured with 4 MB here) and measured a message throughput of around 5.4 GByte/s with 8 MB messages. The throughput is lower as application threads and transport thread work sequentially for larger messages (see Section V-A). However, if the application needs to handle large messages often, throughput can easily be improved by using a larger ORB.

DXNet is designed to efficiently support concurrent application threads sending and receiving messages in parallel. Figure 11 shows that the processing time for 1 KB messages is stable from one to 64 and only slightly increases with 128 application threads. Additionally, Figure 11 shows the performance with a varying number of message handlers peaking with two to four. Obviously, 128 application threads and 128 message handlers overstress the CPU (8 cores) significantly. The results for all other constellations are as expected showing DXNet's capability to efficiently handle hundreds of concurrent threads.

We also evaluated request-response latency by measuring the **RTT**, which includes sending a request, receiving the request, sending the corresponding response and receiving the response. Figure 12 shows the latency for small requests with an increasing number of application threads. The average RTT with one and two application threads is under 5 µs. With up to eight threads the RTT increases slower than the number of threads because requests can be aggregated for sending. With more threads, the increase rate is higher.

Figure 13 shows the breakdown of request-response latency for one and four application threads and 1024-byte requests.

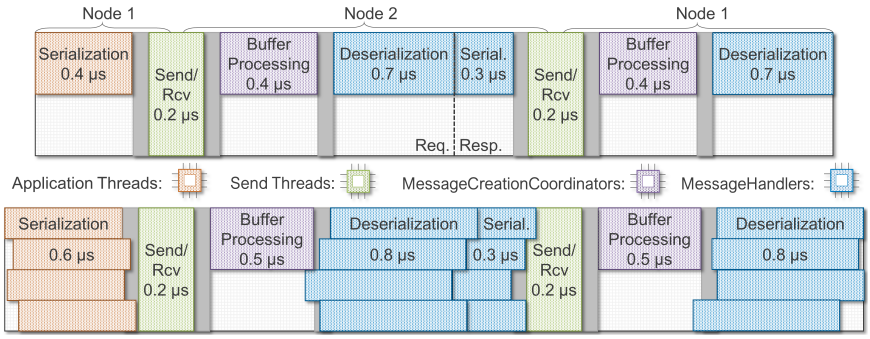

Figure 13. Breakdown of Request-Response Latency for 1024-byte Requests. One application thread (on top) and four (at the bottom). Grey bars indicate inter-thread communication.

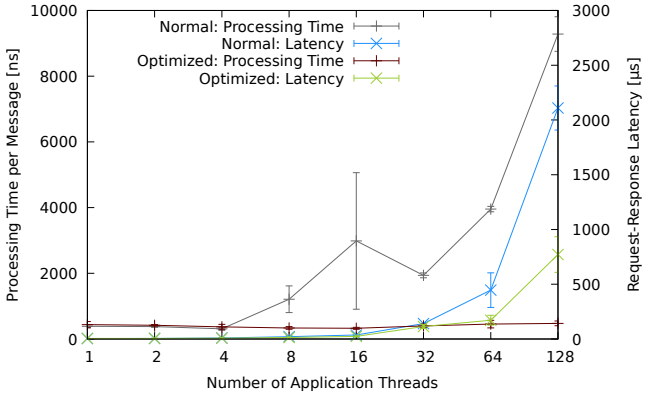

Figure 14. 10^7 Message or 10^6 Requests, 2 Message Handlers, 1 Byte Payload.

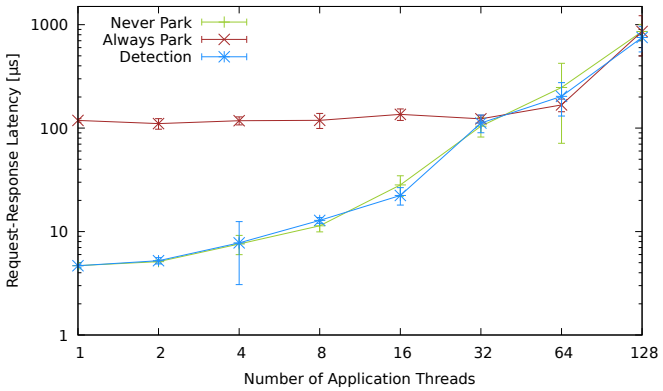

Figure 15. 10^6 Requests, 2 Message Handlers, 1 Byte Payload.

This is a best-effort approximation as time measurement is costly and influences the processing. As expected de-/serialization accounts for the majority of the RTT and deserialization is slower than serialization because of the message object allocation and creation. With more application threads or asynchronous messages, all depicted steps are executed in parallel.

Optimized Outgoing Ring Buffer. The benefits of the Catch-Up Buffer, discussed in Section IV, can be seen in Figure 14. Without the optimization, the message processing time increases significantly with more than four application threads sending messages (with 128 threads nearly 20 times higher). Furthermore, the RTT diverges considerably with more than 32 application threads as well.

Overprovisioning Detection. Figure 15 shows the importance of the thread parking strategy (see Section VII). The RTT is 25 times higher when using one application thread and always parking network threads. All three strategies match with 32 threads and diverge a little with more threads. The never park strategy is at a disadvantage with many threads (128) and the RTT is around 100 µs higher than with the adaptive approach.

The evaluation with Loopback transport shows the high throughput and low latency of DXNet. Furthermore, DXNet provides high stability when used with many threads sending and receiving messages in parallel.

B. *Comparing Network Transports*

Figure 16 shows the message processing time and throughput for all three network transports (Ethernet and Loopback on cluster and cloud instances) with varying payload size. As expected, InfiniBand has the lowest processing overhead and highest throughput of all physical devices.

The comparison between the 1 GBit/s Ethernet of the private cluster and 5 GBit/s Ethernet in Azure cloud reveals interesting insights. Obviously, message throughput is higher in the cloud for large messages. But, message throughput is higher and processing time is lower on the cluster for messages smaller than 64 bytes which is most likely caused by the virtualization overhead of cloud servers. Loopback is also considerably faster on cluster instances (< 300 ns processing time and > 16 GByte/s throughput).

Figure 17 shows the request-response latency and throughput for requests sent by four application threads. Again, 1 GBit/s Ethernet on our cluster performs better for small payloads (< 1024) than 5 GBit/s Ethernet in the cloud. For larger requests, the bandwidth becomes more and more important favoring the cloud network. Both Ethernet networks are far off the latencies InfiniBand achieves. For small request (< 512 byte payload) the RTT is consistently under 10 µs and rises to only 16 µs for 16 KB requests. Hence, the throughput is much higher with InfiniBand as well.

The evaluation with three different physical transports confirms the results gathered with Loopback. DXNet performs strongly especially with InfiniBand (RTT < 10 µs, throughput > 9 GByte/s full-duplex).

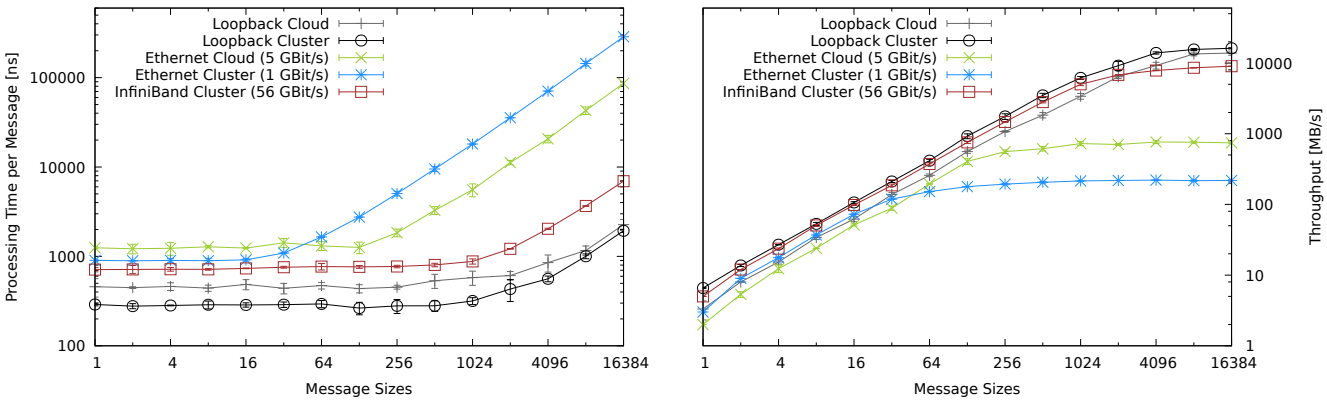

Figure 16. 10^8 Messages, 1 App. Thread, 2 Message Handlers.

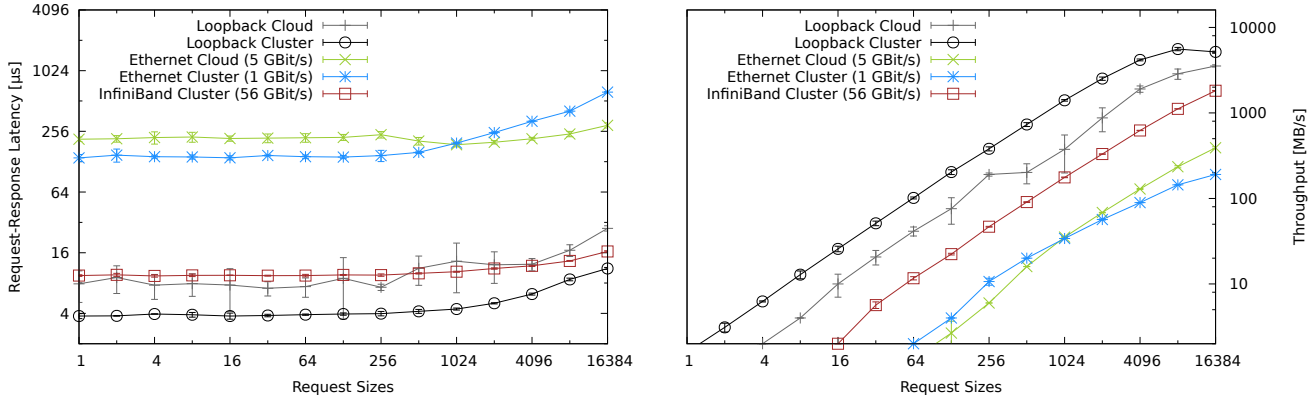

Figure 17. 10^7 Requests, 4 App. Threads, 2 Message Handlers.

TABLE IV. ADDITIONAL PARAMETERS

Parameter	Value
ORB Size	4 MB
Flow Control Windows Size	2 MB
Flow Control Threshold	0.6
net.core.rmem_max	4 MB
net.core.wmem_max	4 MB

C. Scalability of (Eth)DXNet

Message Throughput: First, we measured the asynchronous message throughput with an increasing number of nodes in an all-to-all test with message payloads of 64 and 4096 bytes. For instance, when running the benchmark with 32 nodes, each node sends 25,000,000 64-byte messages to all 31 other nodes and therefore each node has to send and receive 775,000,000 messages in total. Additional network parameters can be found in Table IV.

Figure 18 shows the average payload throughput for single nodes and Figure 19 the aggregated throughput of all nodes.

For 64-byte messages, the payload throughput is between 200 and 260 MB/s for all node numbers, showing a minimal decrease from 2 to 16 nodes. With 4096-byte messages the throughput improves with up to 8 nodes peaking at 1370 MB/s full-duplex bandwidth (5.5 GBit/s uni-directional). With 64 nodes the throughput is still above 5 GBit/s resulting in an aggregated throughput of 83,376 MB/s. The minor decline in both experiments can be explained by an uneven deployment

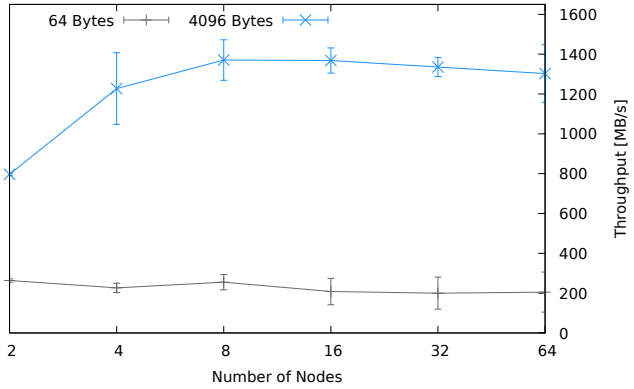

Figure 18. Message Payload Throughput per Node. 1 Application Thread, 2 Message Handler Threads.

of our network benchmark causing the last nodes starting and finishing a few seconds later. The end-to-end throughput between two nodes seems to be bound at around 3.2 GBit/s in the Microsoft Azure cloud as tests with iperf showed, too.

The benchmarks show that DXNet, as well as EthDXNet scale very well for asynchronous messages under high loads.

Request-Response Latency: The next benchmarks are used to evaluate request-response latency by measuring the RTT. Figure 20 shows the RTTs for an all-to-all scenario with 2 to 64 nodes and 1, 16 and 100 application threads. Furthermore, all-to-all tests with `ping` are included to show network latency limitations.

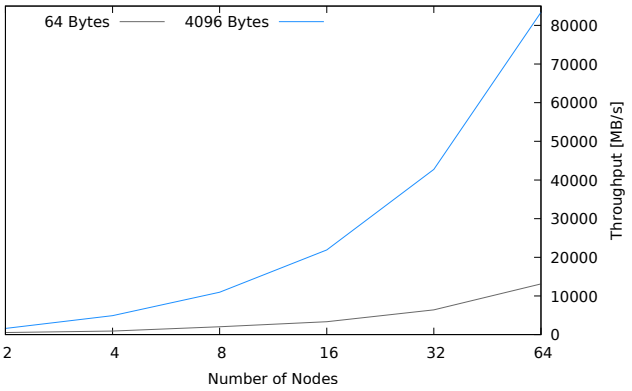

Figure 19. Aggregated Message Payload Throughput. 1 Application Thread, 2 Message Handler Threads.

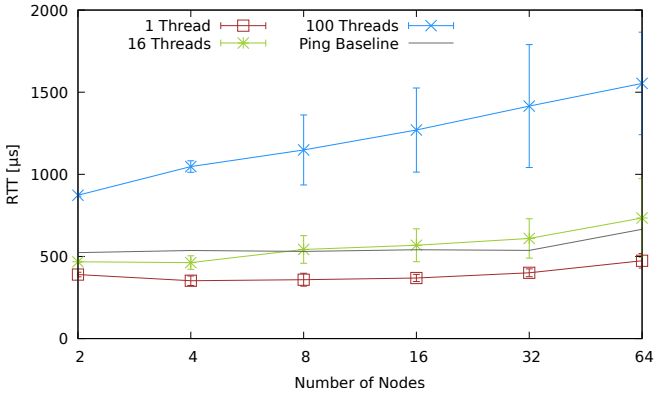

Figure 20. Average Request-Response Latency. 1 to 100 Application Threads, 2 Message Handler Threads.

The latency of the Azure Ethernet network is relatively high with a minimum of 352 µs measured with DXNet and one application thread (Figure 20). A test with up to 4032 ping processes shows that the average latency of the network is even higher (> 500 µs). In DXNet, own requests are combined with responses (and other requests if more than one application thread is used). This reduces the average latency for requests. Additionally, the ping baseline shows an increased latency for more than 32 nodes, by using one scale-set for the first 32 nodes and another one for the last 32 nodes. Different scale-sets are most likely separated by additional switches which increases the latency for communication between scale-sets.

EthDXNet is consistently under the ping baseline demonstrating the low overhead and high scalability of EthDXNet (and DXNet) when using one application thread. With 16 application threads, the latency is slightly higher and on the same level as the baseline, but the throughput is more than 10 times higher as well (in comparison to DXNet with one application thread). Furthermore, both lines have the same bend from 32 to 64 nodes as the baseline.

With 100 application threads per node (up to 6,400 in total), the latency increases noticeably, as expected, because the CPU is highly overprovisioned. In this situation, the latency between writing a message into the ORB and sending it increases dramatically with more open socket channels. Furthermore, requests can be aggregated more efficiently in the ORBs with less open connections masking the overhead with a few nodes.

The latency experiments show that EthDXNet scales up to 64 nodes without impairing latency. With a very high number of application threads (relative to the available cores) the latency increases, as expected, but is still good.

D. Yahoo! Cloud Serving Benchmark

The Yahoo! Cloud Serving Benchmark was designed to quantitatively compare distributed serving storage systems [33]. The benchmark offers a set of simple operations (reads, writes, range scans) and a tabular key-value data model to evaluate online storage systems regarding their elasticity, availability and replication. Furthermore, YCSB is easily extensible for new storage systems and new workloads. For our evaluation, we used the in-memory key-value store DXRAM [8] which

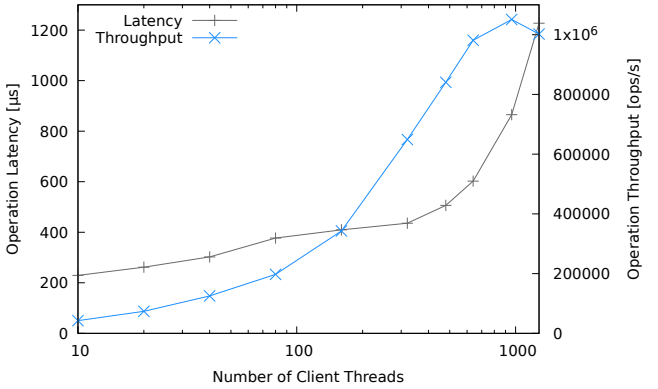

Figure 21. 6 Message Handlers.

utilizes DXNet and created an individual workload: one 64-byte object per key, 10^6 keys, uniform distribution, 90 % read and 10 % write operations, 10^7 operations. The tests were run in the Microsoft Azure cloud with one storage server and an increasing number of client servers (maximum 16) which each hosted up to 80 client threads.

Figure 21 shows the average operation latency and throughput with 10 to 1280 client threads. The operation latency starts at around 230 µs which is in line with previous latency measurements. The latency grows slowly up to 480 client threads but then exponentially indicating server congestion. The throughput rises up to 640 client threads with more than one million operations per second and remains stable with more client threads.

The evaluation with YCSB shows DXNet's high performance for a client-server scenario (one server can serve more than 1000 clients).

XIV. Conclusion and Future Work

Big data applications, as well as large-scale interactive applications, are often implemented in Java and typically executed on many nodes in a cloud data center. Efficient network communication is crucial for these application domains. RMI while being comfortable to use is not fast enough. Plain sockets are difficult to handle especially if efficiency and scalability need to be addressed. MPI was designed for spawning processes with finite runtime in a static environment. Thus, multi-

threading performance and support for adding/removing nodes to an existing environment are limited.

In this paper, we proposed DXNet, a Java open-source network library complementing the communication spectrum. DXNet provides fast parallel serialization for Java objects, automatic connection management, automatic message aggregation and an event-driven message receiving approach including concurrent deserialization. DXNet offers high-throughput asynchronous messaging as well as synchronous request-response communication with very low latency. DXNet achieves high performance and low latency by using lock-free data structures, zero-copy and zero-allocation. The proposed ring buffer and queue structures are complemented by different thread parking strategies guaranteeing low latency by avoiding CPU overload. Finally, its architecture is open for supporting different transport protocols. It already supports TCP with Java.nio and reliable verbs for InfiniBand. We described our practical experiences in designing a transport implementation for Ethernet networks, EthDXNet, integrated into DXNet. EthDXNet provides a double-channel based automatic connection approach using back-channels for sending flow control data and an efficient operation interest handling which is important to achieve low-latency message handling with Java.nio's Selector.

Evaluations on a private cluster and in the Microsoft Azure cloud show message processing times of sub 300 ns resulting in throughputs of up to 16 GByte/s which saturate the memory bandwidth of a typical cloud instance. For the request/response pattern, DXNet is able to provide sub 10 µs RTT latency using the InfiniBand transport (sub 4 µs over Loopback). Finally, DXNet is also able to efficiently handle highly concurrent processing of many small messages resulting in throughput saturations for Ethernet with 256 bytes payload and InfiniBand with 1-2 KB payload. The evaluation in the Microsoft Azure cloud shows the scalability of EthDXNet (together with DXNet) achieving an aggregated throughput of more than 83 GByte/s with 64 nodes connected with 5 GBit/s Ethernet (10 GBit/s Ethernet limited by SLAs). Request-response latency is almost constant for an increasing number of nodes as long as the CPU is not overloaded. Future work includes experiments on larger scales with application traces.

The InfiniBand transport IBDXNet is work in progress and the final results will be published separately (throughput: >10.4 GByte/s). Future work also includes more experiments at larger scales including comparisons with other network middlewares, as well as evaluations using a 100 GBit/s InfiniBand network.

This work has been partially funded by the German DFG.

REFERENCES

[1] K. Beineke, S. Nothaas, and M. Schoettner, "Scalable messaging for java-based cloud applications," *ICNS 2018, The Fourteenth International Conference on Network and Services*, vol. 14, pp. 32–41, May 2018.

[2] A. Ching, S. Edunov, M. Kabiljo, D. Logothetis, and S. Muthukrishnan, "One trillion edges: Graph processing at facebook-scale," *Proc. VLDB Endow.*, vol. 8, pp. 1804–1815, Aug. 2015.

[3] S. Ekanayake, S. Kamburugamuve, and G. C. Fox, "Spidal java: High performance data analytics with java and mpi on large multicore hpc clusters," in *Proceedings of the 24th High Performance Computing Symposium*, 2016, pp. 3:1–3:8.

[4] J. Dean and S. Ghemawat, "Mapreduce: Simplified data processing on large clusters," *Commun. ACM*, vol. 51, pp. 107–113, Jan. 2008.

[5] S. Microsystems, "Java remote method invocation specification," https://docs.oracle.com/javase/7/docs/platform/rmi/spec/rmiTOC.html, accessed: 2018.11.17.

[6] Oracle, "Package java.net," https://docs.oracle.com/javase/8/docs/api/java/net/package-summary.html, accessed: 2018.11.17.

[7] S. Mintchev, "Writing programs in javampi," School of Computer Science, University of Westminster, Tech. Rep. MAN-CSPE-02, Oct. 1997.

[8] K. Beineke, S. Nothaas, and M. Schoettner, "High throughput log-based replication for many small in-memory objects," in *IEEE 22nd International Conference on Parallel and Distributed Systems*, 2016, pp. 535–544.

[9] S. Nothaas, K. Beineke, and M. Schöttner, "Distributed multithreaded breadth-first search on large graphs using dxgraph," in *Proceedings of the First International Workshop on High Performance Graph Data Management and Processing*, ser. HPGDMP '16, 2016, pp. 1–8.

[10] K. Beineke, S. Nothaas, and M. Schoettner, "Dxnet project on github," https://github.com/hhu-bsinfo/dxnet, accessed: 2018.11.17.

[11] W. Zhu, C.-L. Wang, and F. C. M. Lau, "Jessica2: a distributed java virtual machine with transparent thread migration support," in *Proceedings. IEEE International Conference on Cluster Computing*, 2002, pp. 381–388.

[12] R. Noronha and D. K. Panda, "Designing high performance dsm systems using infiniband features," *IEEE International Symposium on Cluster Computing and the Grid, 2004. CCGrid 2004.*, pp. 467–474, 2004.

[13] S. P. Ahuja and R. Quintao, "Performance evaluation of java rmi: A distributed object architecture for internet based applications," in *Proceedings of the 8th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, ser. MASCOTS '00, 2000, pp. 565–569.

[14] M. Philippsen, B. Haumacher, and C. Nester, "More efficient serialization and rmi for java," *Concurrency: Practice and Experience*, vol. 12, pp. 495–518, 2000.

[15] J. Maassen, R. V. Nieuwpoort, R. Veldema, H. Bal, T. Kielmann, C. Jacbos, and R. Hofman, "Efficient java rmi for parallel programming," *ACM Trans. Program. Lang. Syst.*, vol. 23, pp. 747–775, Nov. 2001.

[16] C. Nester, M. Philippsen, and B. Haumacher, "A more efficient rmi for java," in *Proc. of the ACM 1999 Conf. on Java Grande*, 1999, pp. 152–159.

[17] M. P. I. Forum, Ed., *MPI: A Message-passing Interface Standard, Version 3.1 ; June 4, 2015*. High-Performance Computing Center, 2015, 2015. [Online]. Available: https://books.google.de/books?id=Fbv7jwEACAAJ

[18] A. Shafi, B. Carpenter, and M. Baker, "Nested parallelism for multi-core hpc systems using java," in *Journal of Parallel and Distributed Computing*, 2009, pp. 532–545.

[19] M. Baker, B. Carpenter, G. Fox, S. H. Ko, and X. Li, "mpijava: A java interface to mpi," http://www.hpjava.org/mpiJava.html, accessed: 2018.11.17.

[20] G. "Dózsa, S. Kumar, P. Balaji, D. Buntinas, D. Goodell, W. Gropp, J. Ratterman, and R. Thakur, "Enabling concurrent multithreaded mpi communication on multicore petascale systems," in *"Recent Advances in the Message Passing Interface"*, 2010, pp. 11–20.

[21] H. V. Dang, S. Seo, A. Amer, and P. Balaji, "Advanced thread synchronization for multithreaded mpi implementations," in *2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*, May 2017, pp. 314–324.

[22] R. Latham, R. Ross, and R. Thakur, "Can mpi be used for persistent parallel services?" in *Recent Advances in Parallel Virtual Machine and Message Passing Interface*. Springer Berlin Heidelberg, 2006, pp. 275–284.

[23] J. A. Zounmevo, D. Kimpe, R. Ross, and A. Afsahi, "Using mpi in high-performance computing services," in *Proceedings of the 20th European MPI Users' Group Meeting*, ser. EuroMPI '13, 2013, pp. 43–48.

[24] K. Beineke, S. Nothaas, and M. Schoettner, "Fast parallel recovery of many small in-memory objects," in *2017 IEEE 23rd International*

Conference on Parallel and Distributed Systems (ICPADS), Dec. 2017, pp. 248–257.

[25] Oracle, "Java i/o, nio, and nio.2," https://docs.oracle.com/javase/8/docs/technotes/guides/io/index.html, accessed: 2018.11.17.

[26] R. Hitchens, *Java NIO*. Sebastopol, CA, USA: O'Reilly Media, 2009.

[27] G. L. Taboada, J. Touriño, and R. Doallo, "Java fast sockets: Enabling high-speed java communications on high performance clusters," *Comput. Commun.*, vol. 31, pp. 4049–4059, Nov. 2008.

[28] G. L. Taboada, J. Tourino, and R. Doallo, "High performance java sockets for parallel computing on clusters," in *Parallel and Distributed Processing Symposium*, 2007, pp. 1–8.

[29] W. Pugh and J. Spacco, *MPJava: High-Performance Message Passing in Java Using Java.nio*. Springer Berlin Heidelberg, 2004, vol. 16.

[30] L. Mastrangelo, L. Ponzanelli, A. Mocci, M. Lanza, M. Hauswirth, and N. Nystrom, "Use at your own risk: The java unsafe api in the wild," *SIGPLAN Not.*, vol. 50, pp. 695–710, Oct. 2015.

[31] R. Riggs, J. Waldo, A. Wollrath, and K. Bharat, "Pickling state in the javatm system," in *Proc. of the 2nd Conf. on USENIX Conf. on Object-Oriented Technologies*, 1996, pp. 19–19.

[32] "Kryo - java serialization and cloning: fast, efficient, automatic." https://github.com/EsotericSoftware/kryo, accessed: 2018.11.17.

[33] B. F. Cooper, A. Silberstein, E. Tam, R. Ramakrishnan, and R. Sears, "Benchmarking cloud serving systems with ycsb," in *Proc. of the 1st ACM symposium on Cloud computing*, 2010, pp. 143–154.

Health Monitoring and User Profiling for Sports Activities: Evaluating Heart Rate Measurements and Activity Recognition

Toon De Pessemier, Enias Cailliau, and Luc Martens

imec - WAVES - Ghent University
Technologiepark-Zwijnaarde 15
9052 Ghent, Belgium
Email: toon.depessemier@ugent.be enias.cailliau@ugent.be luc1.martens@ugent.be

***Abstract*—Wearables are often equipped with an accelerometer and heart rate sensor. However, the accuracy of the generated heart rate measurements is still unclear. This paper evaluates heart rate measurements during various physical activities performed by test users and compares three types of wearable devices: the specialized sports device with chest strap, the fitness tracker, and the smart watch. Consistent heart rate measurements are obtained with all wearables during activities that require no or very little wrist movements, such as sitting in a chair, cycling, walking, or even squat exercises. In contrast, wearables worn around the wrist (smart watches and fitness trackers) and sport devices worn around the chest measure significantly different heart rates during activities that require a lot of movement of the wrist, such as dumbbell biceps curl and push up exercises. These movements of the user's wrist were measured using the accelerometer of the wearable, and allow the detection of repetitions of a physical activity with a typical movement pattern, such as a dumbbell biceps curl. Based on accelerometer and heart rate data, a user profile is created for a rule-based filter to generate personal recommendations for physical activities. A mobile app demonstrates that heart rate measurements and activity recognition can be used to assist and guide users during workouts.**

Keywords–Activity Recognition; Wearable; Health Information; Recommendation; Personalization.

I. INTRODUCTION

Obesity due to insufficient physical activity is an ever growing problem in modern society [1]. Obesity can induce amongst others, heart diseases and stroke, diabetes, gallbladder disease, and gallstones. Research has shown that the majority of health care costs [2] are directly or indirectly due to physical inactivity [3]. Recent studies in health care support the theory that a healthy diet and regular physical activity are much more effective than traditional medication to cure diabetes [4]. Nutrition and training schedules are online available, but are often not personalized to the user's training goals or physical capacities and are static without taking into account the user's progress.

To tackle this problem, new efforts are made to decrease national obesity levels [5], thereby using technology such as data mining, web frameworks, and multi-modal sensors. Multi-modal sensors enable real-time monitoring of physical activities performed by the user. In the domain of public health monitoring, most of these sensor applications keep track of energy expenditure while performing daily activities [6][7].

Recent wearable devices are often equipped with accelerometers for measuring movements and heart rate sensors. However, the accuracy of heart rate measurements using these devices is still unclear. Manufacturers choose not to assert claims regarding the accuracy of the detection of heart rate patterns; otherwise their gadget would get classified as a medical device and would have to undergo FDA (Food and Drug Administration) regulatory scrutiny [8].

Therefore, this study investigates the accuracy of heart rate measurements performed with different types of wearable devices. The heart rate measurements are evaluated in rest condition, as well as during different physical activities. This paper presents an extension of our previous research on wearables [1]. Compared to this previous study, one additional wearable device, the Polar M600, was evaluated. Moreover, this paper provides a more extensive comparison of the heart rates. The following additional activities are considered: interval training (cycling), rest (sitting in a chair), walking, squats, and push ups. The results of this study are important for (mobile) applications and services that rely on heart rate data generated by these wearables.

Besides heart rate measurements, wearables can perform activity recognition based on the motion detected by the accelerometer. This typically results in a few statistics about the user's physical activity, such as the number of steps taken or the average speed of a running session; but the recognition of specific physical exercises is often still missing. More advanced solutions for activity recognition are often relying on multiple sensors placed on different parts of the body, e.g., on the chest and on the hip, composing a body sensor network [9]. However, this is often considered too intrusive for daily activities. Therefore, this study investigates activity recognition using popular wearable devices (Section VI). More specifically, the number of repetitions of an exercise is counted through activity recognition. Compared to our previous research [1], this papers provides more in depth results by making a distinction between fast execution and slow execution of the exercise.

The goal of this study is to investigate the accuracy of heart rate measurements obtained with different wearables, and to analyze if measurements of heart rate sensor and accelerometer can be combined for an accurate activity recognition. According to our knowledge, this is one of the first studies that compares wearables worn around the wrist and a sports device with a chest strap for heart rate measurements during a physical activity with a lot of movement of the wrist. These

measurement data are the input of recommender systems, which can improve human-web interaction by personalizing interfaces of web applications with tailored suggestions for physical activities. This study presents a rule-based filter as recommender system.

The remainder of this paper is organized as follows. Section II refers to interesting related work. Section III discusses the different methods to measure heart rate. An overview of the wearable devices used in this study is provided in Section IV. The next sections discuss the measurements of the wearables: the heart rate measurements are discussed in Section V, activity recognition is the topic of Section VI, and the usage of the combination of both is covered in Section VII. Section VIII is about the rule-based filter to generate personalized recommendations. Section IX discusses the results and Section X draws conclusions and points to future work.

II. Related Work

The last decade, many digital healthcare services have emerged, also on the mobile platform, to deliver services such as health monitoring, medical consultations, diagnosis, and prescriptions [10][11]. Despite the security and privacy risks [12], more formal and informal health information has become available, with the perspective of a new generation of well-informed, healthy individuals. This phenomenon turns users into health information producers and consumers by offering a multitude of health information services and data [13][14].

To cope with the problem of information overload incurred by this growing availability of data, recommender systems are used as an effective information filter and at the same time as a tool for providing personal suggestions [15][16]. These recommenders may suggest a specific fitness activity or a running trail out of the many available physical activities. But good recommendations should match the physical capabilities of each individual.

To assess the physical load of an activity for a user, measuring the user's physical movements (e.g., using a pedometer) is insufficient, since this neglects the user's effort with respect to his/her physical capacities. The user's physical limits and the intensity of an activity for a user can be estimated by the combination of heart rate measurements and motion sensors [17].

The rising interest in health-related data and applications strengthens the need to monitor heart rate and automatically recognize physical activities on a daily basis. Although the commercial sports devices and wearables are equipped with the necessary hardware to accomplish this challenging task, their accuracy is still unclear.

For commercially available breast belt measuring devices, detailed evaluations of the accuracy have been performed [18]. But for recent wearable devices, only a limited number of studies investigated the accuracy of heart rate data, often in specific conditions. In non-moving conditions, heart rate monitoring using a wrist-worn personal fitness tracker has been evaluated with patients in an intensive care unit [19]. The measured values were slightly lower than those derived from continuous electrocardiographic monitoring, i.e., the medical method for heart rate monitoring. The authors concluded that further evaluation is required to investigate if personal fitness trackers can be used in hospitals, e.g., as early warning systems. Another very related study has investigated the accuracy of step counts and heart rate monitoring with wearables [20]. Test subjects were asked to walk a specific number of steps during the measurements. The accuracy of the heart rate measurements with the tested wearable devices showed to be very high. Our paper contributes to the domain of health monitoring with wearables by studying the accuracy of heart rate measurements during intensive physical activities, and with various types of wearable devices.

In the domain of activity recognition with wearables, the focus is often on the classification of movement or transportation types. Hidden Markov models have been proposed [21] to recognize different physical activities, such as driving a car, riding a bicycle, walking, or standing still. In recent Android versions, similar activity recognition functionality is available through Google's activity recognition API [22].

To classify activities such as walking, race walking, and running based on unlabeled data, an unsupervised method for recognizing physical activities using accelerometers of smart phones has been proposed [23]. Two additional smart phones were attached to the upper arms of the user to recognize specific actions while playing basketball, such as passing or bouncing the ball, or a free throw. Although the accelerometer hardware in smart phones might be very similar for wearables, the specific position of a wearable around the wrist can provide very different data reflecting movements of the hand and wrist, additional to the arm movements.

Other studies investigated how simple actions can be used to recognize more complex activities, which are semantically more representative for a human's real life [24]. The algorithm is based on temporal patterns (such as actions occurring after other actions, or actions that overlap) and a multi-task learning approach [25].

In contrast, our research targets activities that cannot be classified based on the movement speed, but are characterized by specific hand or arm movements, such as dumbbell biceps curl exercises. Our focus is on recognizing the number of repetitions in view of tracking the physical load, rather than on classifying the activities.

The growing availability of these health data on the World Wide Web has brought the problem of information overload [26] to the ehealth domain. For instance, too many sports schedules are available in online databases, but only a minority is matching the physical capabilities and preferences of an individual. This emphasizes the need to personalize health information and services, which is ongoing since the mid-90s [27] and is demonstrated for example in Computer-Tailoring Health Education Systems [26]. Personalization in the health domain is described as "adapting the content of the materials, with the aid of computers, to the specific characteristics of a particular person" [28].

Personalization can be achieved by using a recommender system. Personalized recommendations, tailored messages, and customized information have shown to be far more effective than the non-personalized alternative [15][16].

Health promotion and wellness driven applications often use collaborative filtering techniques to cope with the overload of health data and identify the most relevant information [29].

Collaborative filtering makes a selection of the available information for the user based on actions of the community, and does not rely on a central agency or individual expert. As a result, the quality of the selection is depending on the size and engagement of the community using the service. Alternative content-based solutions do not rely on community activity, but require specific metadata to assess the suitability of information items. Our proposed recommender is a combination of a rule-based and content-based system to filter the content for the users based on their preferences and physical capabilities. Unfortunately, many of the existing recommender systems rely on the manual input of users reporting their performed exercises. In contrast, our solution combines automatic activity recognition and heart rate measurements, which are used as input for the rule-based recommender system.

III. Heart Rate Monitoring

For heart rate monitoring, various methods exist. For this study with wearables, the two most important methods are electrocardiography and photoplethysmography.

Electrocardiography (ECG) is the process of recording the electrical activity of the heart using electrodes placed on the skin [30]. These electrodes detect the small electrical changes on the skin that arise from the heart muscle's electrophysiologic pattern of depolarizing during each heartbeat. For medical purposes, e.g., in hospitals, this technique is applied with 10 electrodes, placed on the patient's limbs and on the surface of the chest.

Photoplethysmography (PPG), also known as optical heart rate sensing, is monitoring heart rate using photo diodes and LEDs [31]. Green light is absorbed by blood, hence its red color. When a light source is covered by a body part (e.g., the wrist in case of a wearable), the light is partially absorbed by the blood and partially reflected. The photo diode captures the reflected light. During a heart beat, more light is absorbed and the photo diode detects a reduction in green light intensity. Although a green LED provides the most accurate results, an infrared LED is often used since this consumes less energy. PPG is a cheap method for measuring heart rate, often used in wearables, but has some disadvantages. Motion artifacts can reduce the accuracy during exercises and free living conditions. Person-dependent variations may also influence the measurements, e.g., a different blood perfusion induces a different absorption of light.

Many wearables are equipped with one or more LEDs to measure heart rate using PPG. The LEDs (light emmitor) and photo diodes (light detector) are typically located on the back of the smart watch or fitness tracker and in direct contact with the skin. The popularity of this technique for wearables is largely due to the convenience and low cost of the hardware. But manufacturers often label their devices as "not a medical device" or "not intended to match medical devices or scientific measurement devices" [32]. This paper discusses the use of PPG in wearables for heart rate measuring in different situations (Section V).

IV. Wearable Devices

For measuring heart rate, three types of wearables were used: a smart watch, a fitness tracker, and a specialized device.

A. Smart Watch

Smart watches are equipped with various sensors but are not medically approved. The smart watch is a general purpose, fashionable device with features such as tracking physical activities and informing users. From a commercial viewpoint, the target group of customers is not limited to sports people, but includes also a broader group of people who like the design or the extra features of the gadget. Smart watches often have hardware capabilities allowing to extend their functionality with additional apps. Smart watches are typically equipped with a light sensor to enable heart rate measurements based on photoplethysmography.

In this study, the *Huawei Watch* was used as smart watch for the measurements because of its popularity and typical smart watch characteristics (e.g., Android Wear). This is a smart watch with a classic design that is not focused on sports activities.

In addition, the *Polar M600* smart watch was used as an alternative wearable device that has a clear focus on tracking sports activities. It is equipped with the typical hardware sensors such as an accelerometer and gyroscope. In contrast to the Huawai watch, the Polar M600 also has a built-in GPS. It is water resistance and can capture swimming metrics. An important characteristic for this research is that optical heart rate measurements are performed with 6 LEDs. These multiple LEDs should improve the accuracy of the heart rate measurements of the Polar M600.

To capture heart rate data in real time, a special Android Wear app was developed for the smart watches. This app communicates with our developed Android app running on a smartphone through the Wearable Data Layer API.

B. Fitness Tracker

These devices, typically worn around the wrist, measure movements and behavior, such as the number of steps taken, sleeping patterns, and sports activities, e.g., a light jog or a mad sprint. As with smart watches, fitness trackers are seldom approved for medical purposes. They are equipped with multiple sensors, such as a 3-axis accelerometer to monitor movement in every direction, an altimeter to measure altitude and keep track of the traveled height, and sometimes a gyroscope to measure orientation and rotation. Compared to smart watches, fitness trackers are more focused on tracking physical activities. In this study, the *Microsoft Band 2* was chosen as fitness tracker because of two reasons. It allows real time analysis of sensor data (heart rate data using photoplethysmography and movement data through the accelerometer) and Microsoft provides a comprehensive API. The API offers functionality, such as aggregating the results of a query, thereby shifting the computational load to the Microsoft servers.

C. Specialized Device

The main purpose of this type of device is measuring heart rate. Typical examples are pulse-oximeters, blood pressure monitors, and heart rate chest straps. These often have only a limited number of sensors and a limited number of features. In this study, the *Polar H7* was used as specialized sports device. This is a popular heart rate chest strap, which produces very accurate measurements (correlation of 0.97 with true heart rate [33]). Heart rate is measured using electrodes in the chest strap that detect heart pulse via an electronic signal.

V. Heart Rate Measurements

To gather, store, and analyze heart rate measurements of these three device types, an Android app was developed and deployed on a Google Nexus 6P smartphone. Figure 1 shows a screenshot of this app. The wearable devices have a Bluetooth communication link with this smartphone and the app has a separate service running for each device to transfer the raw data to the smartphone and store the data in a Realm IO database. Realm is an alternative solution for SQLite and delivers real time performance [34]. The evaluations of the heart rate measurements have been performed in a controlled environment.

Figure 2 shows the data flow through the different components of the system. The data of accelerometers are processed and repetitions of specific movements are detected by activity recognition (Section VI) . The types of activities, the number of repetitions, and the intensity (speed of execution) are used for creating a user profile. Subsequently, a filtering with a rule-based system (Section VIII) is performed to match the intensity of an activity to the physical capabilities of the user (as estimated by the measured heart rate). The best matching activities are offered to the user as personal suggestions.

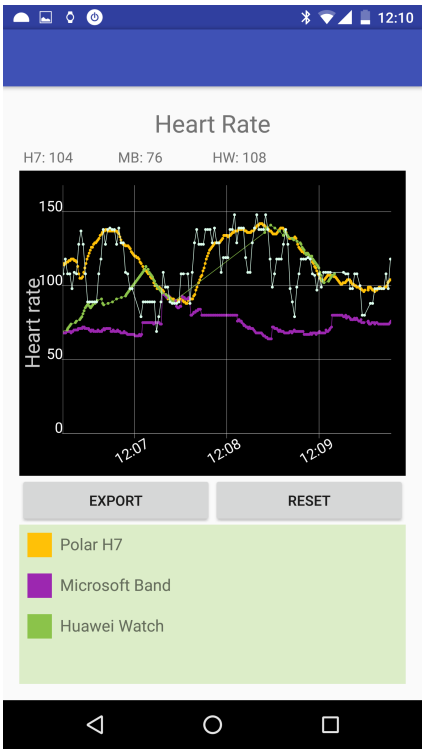

Figure 1. Screenshot of the Android app for gathering heart rate data.

A. In Rest Condition

To evaluate the accuracy of heart rate measurements of the three device types, these heart rate measurements were compared with the measurements of a specialized device that is approved for medical purposes, i.e., the Omrom M6 Comfort [35]. The Omrom M6 is a blood pressure monitor, which has to be attached around the upper arm for measuring the heart rate.

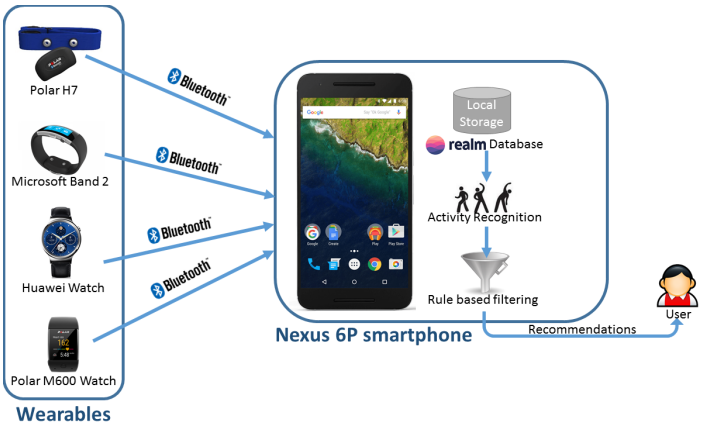

Figure 2. Flow chart showing the data through the different components of the system.

Heart rate was measured for two persons, in a rest condition, in a home environment, at two different times. The first test subject (male) had a low natural heart rate, whereas the second test subject (female) had a rather high heart rate in rest condition. Demographic and physical characteristics have an influence on the absolute value of the heart rate, but this research focuses on comparing measurements of different devices and variations in the heart rate, rather than on the absolute value of the heart rate.

The measurements of the heart rate during rest condition (Table I) were performed in a home, indoor environment. These test users were wearing the four devices simultaneously during the measurements. The Huawei Watch and Microsoft Band were worn around the wrist (one device per wrist), the Polar H7 around the chest, and the Omrom M6 around the upper arm. For optimal measurement results, test users wore only one smart watch per wrist; so the Polar M600 was not used in this experiment. Test users were asked to sit in a chair, doing nothing, while the devices measure their heart rate.

Table I shows for each device the mean, standard deviation, and median, indicating that all devices provide consistent results. The mean values and small standard deviation show that in rest condition, heart rate measurements obtained with these devices can be considered as reliable. The measurements of the Omrom M6, which is medically approved, are considered as the correct heart rate. The measurements of the Polar H7 are the most similar to the measurements of the Omrom M6. Since a blood pressure monitor is rather expensive and not practical during sports activities, the Omrom was not suitable to measure heart rate during physical activities. Therefore, the Polar H7 was considered as the reference device during physical activities.

B. During Dumbbell Biceps Curl

Figure 3 shows the heart rate measurements obtained with the different devices during physical activity, more specifically during the dumbbell biceps curl exercise. This exercise for bicep muscles was performed in a fitness room by two people. Similar results are obtained for both persons (results are shown for only one person). Before each experiment, a rest period of 10 minutes was imposed to avoid influence of previous activities and the coupled heart rate.

TABLE I. MEAN $\bar{x}$, STANDARD DEVIATION σ, AND MEDIAN $\tilde{x}$ OF THE HEART RATE IN REST CONDITION WITH TWO USERS AT TWO TIMES

	User 1 - Test 1		User 1 - Test 2		User 2 - Test 1		User 2 - Test 2	
Device	$\bar{x} \pm \sigma$	$\tilde{x}$	$\bar{x} \pm \sigma$	$\tilde{x}$	$\bar{x} \pm \sigma$	$\tilde{x}$	$\bar{x} \pm \sigma$	$\tilde{x}$
Smart Watch (Huawei Watch)	55±2.0	55	55±2.0	56	73±3.3	73	72±3.2	71
Fitness Tracker (Microsoft Band)	50±2.9	50	64±6.0	64	75±3.3	75	76±1.7	76
Specialized Sports Device (Polar H7)	56±1.7	56	59±1.4	59	77±3.0	76	80±3.7	79
Specialized Blood Pressure Monitor (Omrom M6)	55±2.8	55	58±2.9	58	76±2.5	76	84±4.2	84

During physical activities, such as dumbbell biceps curl, measuring heart rate cannot be performed with the blood pressure monitor due to body movements and the non-wearable characteristic of the Omrom M6. Since the user was wearing the Microsoft Band 2 on one wrist and the Huawai Watch on the other, the Polar M600 smart watch was not evaluated in this experiment. For the three wearable devices that were used (Polar H7, Microsoft Band 2, and Huawai Watch), a significantly different measurement signal of the heart rate can be witnessed during this physical activity.

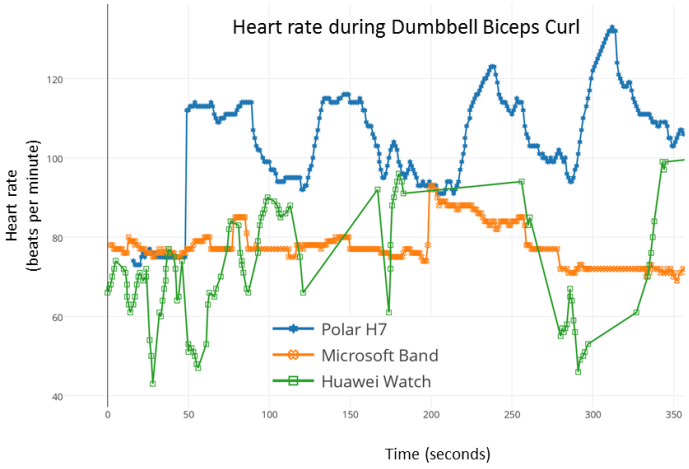

Figure 3. Heart rate measurements during dumbbell biceps curl.

The heart rate signal produced by the *Polar H7* clearly shows a repetitive pattern that corresponds to the repetitions of the dumbbell biceps curl exercise. The accurate measurements can be explained by the use of the chest strap, which is less influenced by movements than the devices worn around the wrist.

The heart rate registered by the *Microsoft Band 2* is consistently lower than the values measured by the Polar device. Moreover, rapidly varying heart rates due to periods of intensive physical activity are difficult to detect. As a result, the subsequent repetitions of the physical exercise are not clearly visible in the graph of the Microsoft Band in Figure 3.

With the *Huawei Watch*, less measurement samples are obtained compared to the Polar H7 and the Microsoft Band. Movements of this device, which is worn around the wrist, cause interruptions in the measurement process. Changes in the device's position relative to the wrist induce a sensor recalibration and can be noticed in Figure 3 as the time periods without measurement data from the Huawei Watch. Periods of intensive physical activities are noticeable by the variations in the data of the heart rate measurements. But the interruptions in the measurement data might be a problem for detailed heart rate monitoring during physical activities.

C. During various Physical Activities

Based on the results of our previous study [1], this section further investigates the differences in the heart rate measurements between the smart watch and a specialized device, such as the Polar H7, as witnessed in Section V-B. For this experiment, heart rate measurements obtained with the Polar H7 were compared to the measurements of the Polar M600. Since, the Polar M600 smart watch is designed for sports activities, the measurements are performed during different physical activities. Special attention is paid to the activities that involve a lot of movement of the wrist in contrast to activities with limited movement of hands and arms.

Although both devices are connected to the same smartphone, as illustrated in Figure 2, a comparison of the raw data shows that the measurements are not synchronous. This can be illustrated by Figure 4, which shows the heart rate measurements during interval training. In this test setup, periods of intensive physical activity and rest periods alternate, which is reflected in a rising and falling heart rate. Figure 4 indicates a lower responsiveness of the Polar M600, which measures heart rate at the wrist, compared to the Polar H7, which measures heart rate at the chest. Sudden increases or decreases in heart rate are only visible a few seconds later with the Polar M600. This is important to take into account for users or application developers that intent to use these data for monitoring activities with rapidly varying heart rate.

By shifting the measurements of the Polar M600 with 9 seconds in time (by subtracting 9 seconds from the timestamp), the measurements appear to be synchronous with the measurements of the H7 chest strap. Figure 5 shows the measurements of both devices after this time shift. To compare the values of the heart rate measurements of both devices during various physical activities, this time shift is applied to all subsequent graphs visualizing heart rate (Figures 5 - 9).

Figure 6 shows the heart rate measurements during rest. The test users are sitting in a chair doing nothing, while wearing both devices to measure the heart rate for a period of 5 minutes. During this period, the measured heart rate varies between around 55 and 70 beats per minute (bpm). The measurements of both devices are very similar, except around second 120 and 195, differences up to 12 bpm are witnessed. This discrepancy was not due to movement of the wrist or body of the user, but might be due to inaccuracies of the measurement process. However, for most of the measurement period, the differences between both devices are small.

Table II lists the mean difference (Mean diff), the mean absolute difference (Mean abs diff), the median absolute difference (Median abs diff) and the maximum absolute difference (Max abs diff) between the measurements with the two devices.

The low values of the mean difference might be an indication that both devices report a reliable heart rate. The

measurements of one of the devices are not consistently higher than the ones of the other device. The mean absolute difference is a measure for the average error between both. The low value in rest condition shows that measurements are very accurate. The maximum absolute difference stands for the largest error between both measurements. In rest condition, this error is obtained around second 195. The median absolute error is less influences by these extreme values. In rest condition, the typical difference between both devices is only 1 bpm.

Figure 7 shows the heart rate measurements while doing a light activity. In this test setup, the users were asked to walk while measuring heart rate. Users were is rest condition (sitting) before the test and during the first 30 seconds, so that an increase in heart rate could be detected during the first minutes. At timestamp 30, users started walking for 7 minutes. Subsequently, the users returned to their seat (rest condition), which can be seen by the decreasing heart rate during the last 3 minutes. Compared to rest condition, the differences in Table II are slightly higher for walking, but still small.

Figure 5 shows the heart rate during interval training. In this test setup, the accuracy is evaluated in case of peaks in the heart rate. The users were first in rest condition for 30 seconds. Next, the users were cycling on a home trainer during 1 minute, followed by 1 minute of rest. Subsequently, this pattern of 1 minute cycling and 1 minute resting was done 2 times more (so, 3 times in total). Again, Table II shows small differences between the measurements of both devices.

In the last two test setups, test users were performing muscle strengthening activities. This way, the accuracy of the heart rate can be evaluated during intensive physical activities, which are characterized by many body movements and a rapid variation of the heart rate.

Figure 8 shows the measurements obtained while performing squats. The squat is a compound, full body exercise that trains multiple types of muscles. For this, users have to bend the knees and move their arms, but wrist movements are limited. A similar test setup is used here. During the first 30 seconds, the users were doing nothing in order to measure their heart rate in rest. Next, the users were executing squat exercises during 30 seconds, followed by 30 seconds of rest. Subsequently, the users were asked to do a new set of squats during 30 seconds, rest 30 seconds, doing a last set of squats during 30 seconds, and rest 30 seconds. Although this exercise involves a lot of movement with arms and legs, thereby increasing the heart rate rapidly, the measurement differences remain small, as shown in Table II.

Figure 9 shows the heart rate measurements while doing push ups. The push up exercise is performed in a prone position by raising and lowering the body using the arms. It involves a lot of pressure on the wrists and subsequent repetitions implicate some movement of the wrists. Again, the test started with a period of 30 seconds in rest condition. Subsequently, the user performs push ups during 30 seconds, followed by 30 seconds of rest. This pattern (30 seconds push ups, 30 seconds rest) is executed 3 times.

For this activity, Table II and Figure 9 show large differences between the two different devices. These differences can be explained by the movements of the wrist, which are characteristic for this activity and much more prominent compared to the other activities, such as squats or interval training

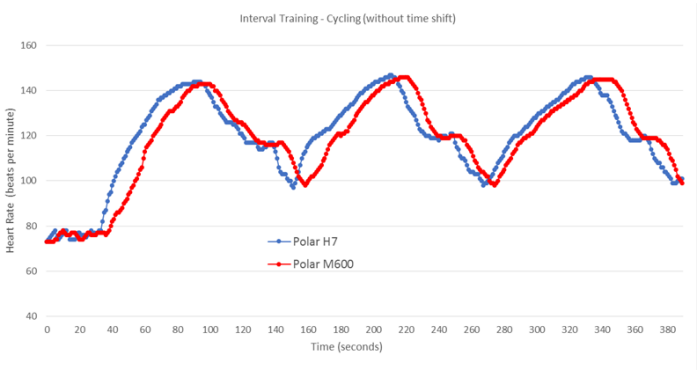

Figure 4. Heart rate measurements during interval training without a time shift.

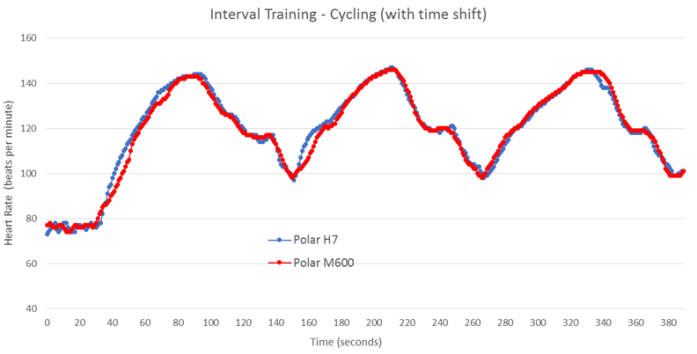

Figure 5. Heart rate measurements during interval training with a time shift.

(cycling). The range of the heart rate is similar for interval training, squats, and push ups, starting at approximately 70 bpm and increasing until 120 - 140 bpm. For interval training and squats, the differences between the Polar H7 and M600 are typically only 1 bpm (median value), whereas for push ups the median difference is 18 bpm with maximum differences up to 58 bpm. Figure 9 shows that the two devices provide consistent measurement values during rest periods: during the first and last 30 seconds of the test, the heart rates of both devices are very similar. In contrast, during periods of push up activities, the two devices output very different measurement results. The Polar H7 shows a heart rate increase with peaks up to 130 bpm and active periods can be clearly detected, whereas the M600 measures only a slight heart rate increase up to 100 bpm and active periods can hardly be distinguished. This confirms the hypothesis that heart rate measurements with smart watches are less accurate during physical activities that involve a lot of movement of the wrist, such as push ups or dumbbell biceps curl exercises.

TABLE II. COMPARATIVE STATISTICS FOR THE HEART RATE MEASUREMENTS OF THE POLAR H7 AND POLAR M600.

Activity	*Mean diff*	*Mean abs diff*	*Median abs diff*	*Max abs diff*
Rest	0.36	1.40	1	12
Walking	0.75	1.48	1	18
Interval	0.61	1.88	1	9
Squats	1.32	2.11	1	15
Push Ups	19.39	19.56	18	58

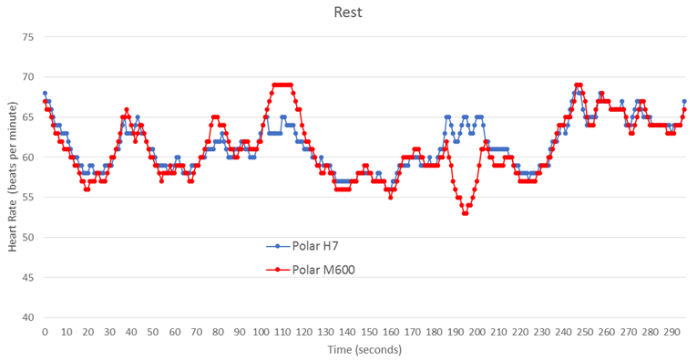

Figure 6. Heart rate measurements during rest.

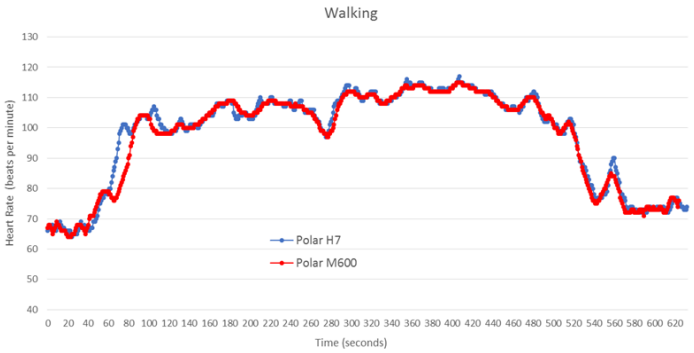

Figure 7. Heart rate measurements during walking.

Figure 8. Heart rate measurements during squats.

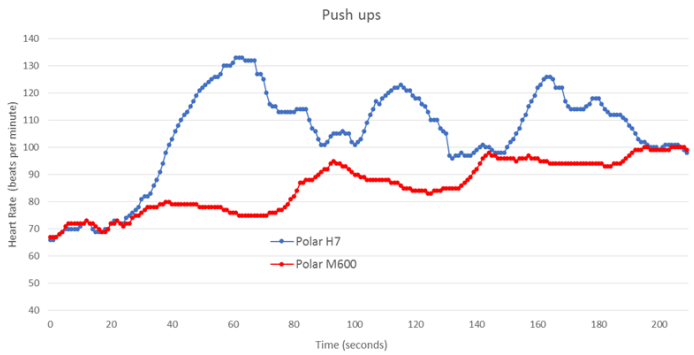

Figure 9. Heart rate measurements during push ups.

VI. Activity Recognition

Many studies have tackled the challenge of activity recognition by using multiple sensors placed on different parts of the body, e.g., on the hip and the chest [9]. But, the placement of these sensors can be considered too expensive and invasive for daily sports activities. In contrast, this study performs activity recognition using wearables.

The aim is to assist users in coaching tasks such as checking the proper execution of physical exercises or counting the number of repetitions of an exercise, rather than classifying the physical activities by type. Repetitions of an exercise are detected by real-time processing of the raw data of the accelerometer of the wearable worn around the wrist. The dumbbell biceps curl exercise is a typical activity that allows detection of repetitions of this exercise by using data of the accelerometer. This exercise is characterized by specific movements of the arm and wrist, which enable the detection.

Figures 10 and 11 show the pattern of the accelerometer data, gathered with the fitness tracker around the wrist, during the execution of this exercise. As an extension to our previous research [1], the accelerometer data was gathered during fast execution as well as slow execution. Figure 10 visualizes the accelerometer data gathered during a fast execution of the exercise. A slow execution, with a clearer pattern, is visible in Figure 11. The comparison of Figures 10 and 11 demonstrates that higher maximum and lower minimum values are reached with a fast execution. Although the execution speed of the activity and the body characteristics of the user may have an influence on the absolute values of the data of the accelerometer, the typical pattern consisting of local minimums and maximums can be witnessed for every repetition.

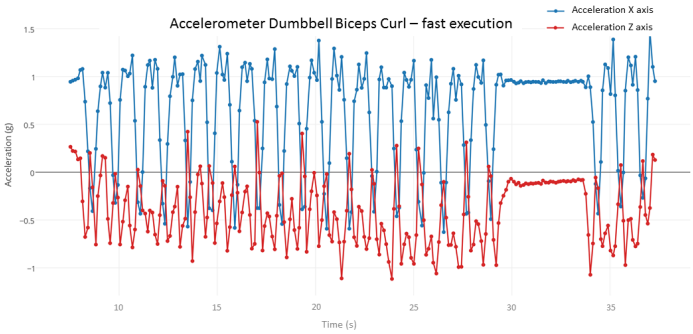

Figure 10. Measurements of the accelerometer of a wearable during fast execution of a dumbbell biceps curl exercise.

For each repetition of the dumbbell curl activity, 5 local optima on the Z-axis and 3 on the X-axis can be witnessed as visible in Figure 12: 1) a maximum on the Z-axis co-occurring with a maximum on the X-axis, 2) a minimum on the Z-axis, 3) a maximum on the Z-axis co-occurring with a minimum on the X-axis, 4) a minimum on the Z-axis, and 5) a maximum on the Z-axis co-occurring with a maximum on the X-axis. The red cross marks in Figure 12 denote the beginning and end of a repetition of the exercise, the green check marks indicate the intermediate optima. Recognizing the dumbbell biceps curl execution based on the identification of a sequence of these 5 events has some benefits. The recognition process requires limited processing power, allowing real-time recognition (e.g., for e-coaching purposes) and making it usable on devices

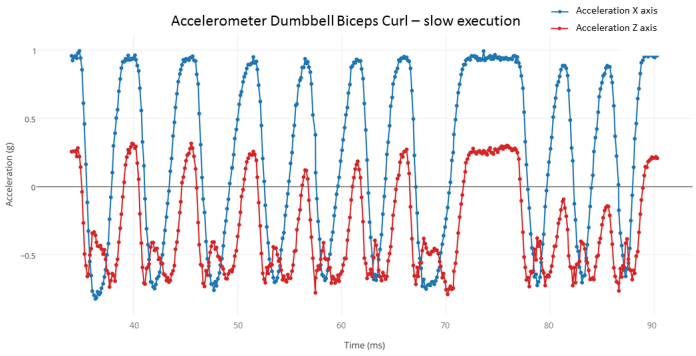

Figure 11. Measurements of the accelerometer of a wearable during slow execution of a dumbbell biceps curl exercise.

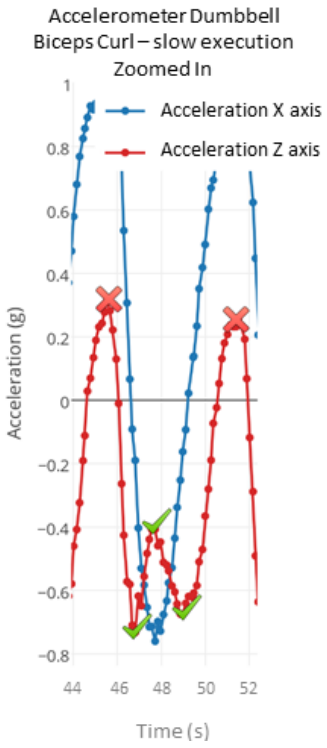

Figure 12. Detailed view on the local optima in the accelerometer data during slow execution of a dumbbell biceps curl exercise.

with limited processing power, such as wearables. Moreover, the detection of local optima makes the recognition method directly usable for different variations of the dumbbell curl, such as concentration curl, hammer curl, and barbell curl.

VII. Heart Rate and Activity Recognition Combined

Monitoring heart rate and simultaneously recognizing repetitions of an activity with the accelerometer allow a better health monitoring and e-coaching during workouts. Since raw data streams of both sources (heart rate sensor and accelerometer) are suffering from inaccuracies, the combination of both can improve health monitoring. For example, the intensity of a physical activity for an individual can be estimated based on the heart rate data. But in case of measurement interruptions in the heart rate data, accelerometer data can be used to estimate the performed physical activities.

For e-coaching purposes, our Android app uses both data sources to instruct the user during physical exercises thereby maintaining a healthy heart rate. Repetitions of an exercise are recognized and through text-to-speech techniques the repetitions are counted aloud or shown on the screen of the wearable. Each physical activity has a target range of the heart rate that can be expected during the execution. If the measured heart rate is out of this range, the user is warned by a clear indication on the screen of the wearable. After performing an activity, the app evaluates the intensity of the physical exercise as “too intensive”, “to easy”, or “just good”.

VIII. User Profiling and Recommendations

The physical exercises measured with the accelerometer, the heart rate, and the characteristics of the exercises are stored online in a user profile. Users can access their profile using a web application to analyze their history of physical activities. Moreover, this user profile is used for personalization of suggestions for new activities in our Android app, such as a set of dumbbell biceps curl exercises, a running track, a cycling track, etc. Figure 13 shows a screenshot of the recommended activities for one of the users.

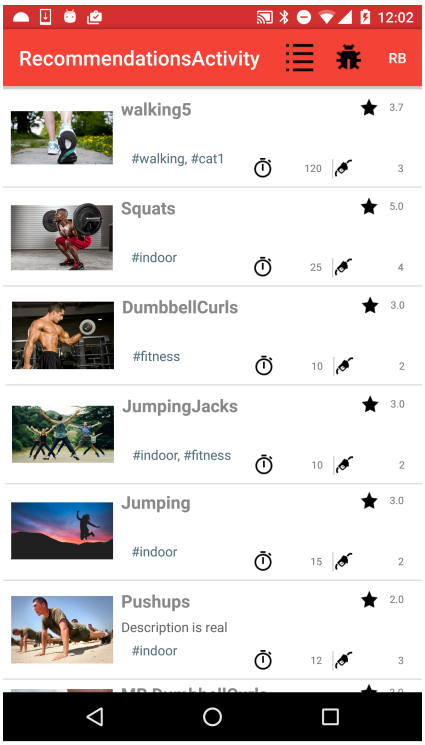

Figure 13. Screenshot of the Android app showing the recommended activities.

To match the user’s preferences and physical capabilities to the physical activities and select the most suitable ones as recommendations, the activities of each type are processed by a specialized rule-based filter. This rule-based filter makes a selection of the activities based on characteristics of that type of activity, e.g., the distance for a running activity or the weight and number of repetitions for dumbbell biceps

curl. For each type of activity, a separate rule-based filter is used in order to take into account the user's experience level for each activity individually. For example, suppose a user is an excellent runner. Recommendations for intensive running activities will be the most suitable, given the user's physical capabilities and history. Now, suppose that this user visits the gym for the first time with the goal of training the arm muscles. The user's body is not used to intensive dumbbell biceps curl activities. Recommendations at the level of starting users might be appropriate here. Therefore, a separate rule-based filter is assigned to each activity type to handle these differences in training level for users. In future work, explanations about the recommendations can be added in order to further convince the user to adopt one of the offered recommendations [36]. These explanations can be expressed in terms of (the progress of) the physical capabilities of the user.

The rule-based functionality is implemented based on Drools [37]. Drools is a business rules management system with business rules engine that is scalable and extendible through the use of drl files containing the rules. The goal of these rules is to filter the available activities in order to come up with the most suitable activity for the user taking into account the conditions/context at the moment of the recommendation.

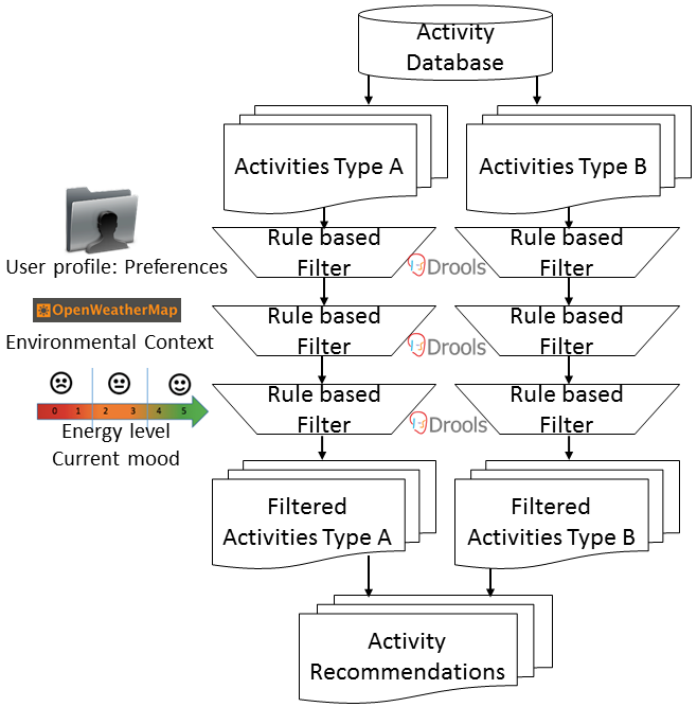

Figure 14. Graphical overview of the rule-based filtering of the activities.

Figure 14 gives a graphical overview of the rule-based filtering that is applied to the activities. The rules check the following conditions: 1) *User profile*: Do the length and intensity of the cycling or running track match the user's physical capabilities and habits? The target length of a track is similar to the length of the tracks in the user's history, but a small difference is tolerated. The intensity of the track is estimated based on the difference in altitude meters. For gym exercises, the intensity is estimated based on the weight or resistance of the fitness equipment and the number of repetitions. 2) *Environmental Context*: Does the activity match the current weather conditions? For example, outdoor running activities are not recommended when it rains. To retrieve weather data at the user's location, the OpenWeatherMap.org REST API [38] is used. 3) *Energy Level*: Does the activity match the user's energy level of the moment? The energy level is a value reflecting the user's current energy mood, ranging from 0 to 5, that users can specify in the Android app to express their current feeling, e.g., energetic, tired, or something in between.

IX. Discussion

During intensive physical activities with limited movement of the wrist, such as walking, cycling, or even squat exercises, the experiments showed accurate heart rate measurements (Section V-C). In contrast, a discrepancy in the measurements of the wearables is witnessed during intensive physical activities with a lot of wrist movements (dumbbell biceps curl and push ups). Shifts of the wearable with respect to the position of the wrist induce inaccuracies or even interruptions in the measurement process thereby hindering the monitoring of heart rate variations. The resulting measurements obtained with wearables around the wrist are often lower than the heart rate measured with a specialized sports device.

Specialized sports devices, using a sensor with chest strap, produce more accurate heart rate measurements, even during intensive physical activities, and enable recognizing subsequent repetitions of a physical activity based on the periodic peaks in the heart rate. Therefore, our advise is to use a device with a chest strap for heart rate measurements in case of physical activities that involve a lot of movement of the wrist.

Besides, raw data produced by the accelerometer of wearables can be used to recognize repetitions of physical exercises with characteristic movements of the wrist/hand/arm. E.g., the dumbbell biceps curl exercise can be recognized based on a specific pattern with 5 local optima on the X and Z-axis of accelerometer data. Both raw data streams (heart rate and accelerometer data) can be combined for further analysis, but also to assist the user in coaching tasks, such as counting the number of times an exercise is performed, or instructing to decrease or increase the intensity of the exercise. Automatic activity recognition can help the user by reducing the burden of providing input about the performed activities in digital health services or fitness apps. Moreover, recognized activities can be stored in a user profile, which can be used as an indicator for the user's physical capabilities, habits or preferences for sports activities. Based on this profile, the current weather, and the user's mood (energy level as specified by the user), personalized recommendations are generated using a set of rule-based filters.

X. Conclusion

This study discussed the usage of wearables for heart rate measurements and the automatic recognition of physical activities. Measurements with a fitness tracker and a smart watch showed to be very accurate in case of limited physical movement, e.g., in a state of rest. During intensive physical activities with a lot of wrist movements, measurements performed with wearables might be disturbed. Simultaneously

measured accelerometer data showed to be useful for recognizing repetitions of physical exercises with characteristic movements of the arm, hand, or wrist. The combination of heart rate measurements and activity recognition allows to create a use profile that reflects the user's physical capabilities in view of generating personal recommendations. In future research, we will investigate the recognition of other physical exercises and relate the resulting accelerometer data to heart rate data more in depth.

REFERENCES

[1] T. D. Pessemier, E. Cailliau, and L. Martens, "Heart rate monitoring and activity recognition using wearables," in Proceedings of the Sixth International Conference on Building and Exploring Web Based Environments (WEB2018), May 2018, pp. 1–6.

[2] T. Bodenheimer, E. H. Wagner, and K. Grumbach, "Improving primary care for patients with chronic illness," Jama, vol. 288, no. 14, 2002, pp. 1775–1779.

[3] F. W. Booth, C. K. Roberts, and M. J. Laye, "Lack of exercise is a major cause of chronic diseases," Comprehensive Physiology, vol. 2, no. 2, 2012, p. 1143.

[4] S. Hammer, J. Kim, and E. André, "Med-styler: Metabo diabeteslifestyle recommender," in Proceedings of the Fourth ACM Conference on Recommender Systems, ser. RecSys '10. New York, NY, USA: ACM, 2010, pp. 285–288.

[5] S. A. Khan et al., "Gethealthyharlem.org: developing a web platform for health promotion and wellness driven by and for the harlem community." AMIA. Annual Symposium proceedings / AMIA Symposium., vol. 2009, 2009, pp. 317–321.

[6] S. Chen, J. Lach, O. Amft, M. Altini, and J. Penders, "Unsupervised activity clustering to estimate energy expenditure with a single body sensor," in 2013 IEEE International Conference on Body Sensor Networks, May 2013, pp. 1–6.

[7] F. Dadashi et al., "A hidden markov model of the breaststroke swimming temporal phases using wearable inertial measurement units," in 2013 IEEE International Conference on Body Sensor Networks, May 2013, pp. 1–6.

[8] J. Kim, "Wearable heart rate monitors enter the consumer mainstream," 2014, [Online] Retrieved March, 2018, from http://internetofthingsagenda.techtarget.com/feature/Wearable-device-heart-rate-monitoring-entering-the-consumer-mainstream.

[9] G. Plasqui, A. Bonomi, and K. Westerterp, "Daily physical activity assessment with accelerometers: new insights and validation studies," Obesity Reviews, vol. 14, no. 6, 2013, pp. 451–462.

[10] D. G. Korzun, A. Y. Meigal, A. V. Borodin, and L. I. Gerasimova-Meigal, "On mobile personalized healthcare services for human involvement into prevention, therapy, mutual support, and social rehabilitation," in 2017 International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON). IEEE, 2017, pp. 276–281.

[11] Y. V. Zavyalova, D. G. Korzun, A. Y. Meigal, and A. V. Borodin, "Towards the development of smart spaces-based socio-cyber-medicine systems," International Journal of Embedded and Real-Time Communication Systems (IJERTCS), vol. 8, no. 1, Jan. 2017, pp. 45–63.

[12] G. Kambourakis, E. Klaoudatou, and S. Gritzalis, "Securing medical sensor environments: the codeblue framework case," in The Second International Conference on Availability, Reliability and Security, 2007. ARES 2007. IEEE, 2007, pp. 637–643.

[13] G. Eysenbach, "Consumer health informatics," British medical journal, vol. 320, no. 7251, 2000, p. 1713.

[14] R. J. Cline and K. M. Haynes, "Consumer health information seeking on the internet: the state of the art," Health education research, vol. 16, no. 6, 2001, pp. 671–692.

[15] V. J. Strecher et al., "The effects of computer-tailored smoking cessation messages in family practice settings," Journal of Family Practice, vol. 39, no. 3, 1994, pp. 262–270.

[16] M. K. Campbell et al., "Improving dietary behavior: the effectiveness of tailored messages in primary care settings." American journal of public health, vol. 84, no. 5, 1994, pp. 783–787.

[17] P. S. Freedson and K. Miller, "Objective monitoring of physical activity using motion sensors and heart rate," Research quarterly for exercise and sport, vol. 71, no. sup2, 2000, pp. 21–29.

[18] M. Weippert et al., "Comparison of three mobile devices for measuring r–r intervals and heart rate variability: Polar s810i, suunto t6 and an ambulatory ecg system," European journal of applied physiology, vol. 109, no. 4, 2010, pp. 779–786.

[19] R. R. Kroll, J. G. Boyd, and D. M. Maslove, "Accuracy of a wrist-worn wearable device for monitoring heart rates in hospital inpatients: A prospective observational study," Journal of medical Internet research, vol. 18, no. 9, 2016, p. 253.

[20] F. El-Amrawy and M. I. Nounou, "Are currently available wearable devices for activity tracking and heart rate monitoring accurate, precise, and medically beneficial?" Healthcare informatics research, vol. 21, no. 4, 2015, pp. 315–320.

[21] J. Lester, T. Choudhury, N. Kern, G. Borriello, and B. Hannaford, "A hybrid discriminative/generative approach for modeling human activities," in Proceedings of the 19th International Joint Conference on Artificial intelligence. Morgan Kaufmann Publishers Inc., 2005, pp. 766–772.

[22] Google, "Activity Recognition API," 2018, [Online] Retrieved March, 2018, from https://developers.google.com/android/reference/com/google/android/gms/location/ActivityRecognitionApi.

[23] Y. Lu, Y. Wei, L. Liu, J. Zhong, L. Sun, and Y. Liu, "Towards unsupervised physical activity recognition using smartphone accelerometers," Multimedia Tools and Applications, vol. 76, no. 8, Apr 2017, pp. 10 701–10 719.

[24] Y. Liu, L. Nie, L. Han, L. Zhang, and D. S. Rosenblum, "Action2activity: Recognizing complex activities from sensor data." in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), 2015, pp. 1617–1623.

[25] Y. Liu, L. Nie, L. Liu, and D. S. Rosenblum, "From action to activity: Sensor-based activity recognition," Neurocomputing, vol. 181, no. Supplement C, 2016, pp. 108 – 115, big Data Driven Intelligent Transportation Systems.

[26] L. Fernandez-Luque, R. Karlsen, and L. Vognild, "Challenges and opportunities of using recommender systems for personalized health education." Studies in health technology and informatics, vol. 150, 2008, pp. 903–907.

[27] K. Binsted, A. Cawsey, and R. Jones, Generating personalised patient information using the medical record. Berlin, Heidelberg: Springer Berlin Heidelberg, 1995, pp. 29–41.

[28] H. de Vries and J. Brug, "Computer-tailored interventions motivating people to adopt health promoting behaviours: Introduction to a new approach," Patient Education and Counseling, vol. 36, no. 2, 1999, pp. 99–105.

[29] D. Jannach, M. Zanker, A. Felfernig, and G. Friedrich, Recommender Systems: An Introduction, 1st ed. New York, NY, USA: Cambridge University Press, 2010.

[30] L. N. Katz and A. Pick, Clinical electrocardiography. Lea & Febiger, 1956.

[31] A. Reisner, P. A. Shaltis, D. McCombie, and H. H. Asada, "Utility of the photoplethysmogram in circulatory monitoring," The Journal of the American Society of Anesthesiologists, vol. 108, no. 5, 2008, pp. 950–958.

[32] T. Collins, I. M. Pires, S. Oniani, and S. Woolley, "How reliable is your wearable heart-rate monitor?" 2018, [Online] Retrieved November, 2018, from https://medicalxpress.com/news/2018-06-reliable-wearable-heart-rate.html.

[33] M. Altini, "Heart Rate Variability for Training," 2013, [Online] Retrieved March, 2018, from http://www.marcoaltini.com/blog/heart-rate-variability.

[34] Realm, "Realm - create reactive mobile apps in a fraction of time," 2018, [Online] Retrieved March, 2018, from https://realm.io/.

[35] Omron, "Automatic Blood Pressure Monitor - Model M6 Comfort IT," 2015, [Online] Retrieved March, 2018, from http://www.omron-healthcare.com/en/support/manuals/download/m6-comfort-it-hem-7322u-e-en.

[36] N. Karacapilidis, S. Malefaki, and A. Charissiadis, "A novel framework for augmenting the quality of explanations in recommender systems," Intelligent Decision Technologies, vol. 11, no. 2, 2017, pp. 187–197.

[37] J. Community, “Drools - Business rules management System,” 2018, [Online] Retrieved March, 2018, from http://www.drools.org/.

[38] OpenWeatherMap, “Current weather and forecast,” 2018, [Online] Retrieved March, 2018, from http://openweathermap.org/.

www.iariajournals.org

International Journal On Advances in Intelligent Systems
issn: 1942-2679

International Journal On Advances in Internet Technology
issn: 1942-2652

International Journal On Advances in Life Sciences
issn: 1942-2660

International Journal On Advances in Networks and Services
issn: 1942-2644

International Journal On Advances in Security
issn: 1942-2636

International Journal On Advances in Software
issn: 1942-2628

International Journal On Advances in Systems and Measurements
issn: 1942-261x

International Journal On Advances in Telecommunications
issn: 1942-2601