Universität
Rostock
Traditio et Innovatio

Network and Application Design for Reliable Distributed Systems

Habilitation thesis
submitted
in partial fulfillment of the requirements for the degree
Doktor-Ingenieur habilitatus (Dr.-Ing. habil.)
of the Faculty of Computer Science and Electrical Engineering
of the University of Rostock

submitted by
Danielis, Peter, born on May 19, 1982 in Wismar, Germany
from Rostock, Germany

Rostock, Germany, June 1, 2019

https://doi.org/10.18453/rosdok_id00002493

Reviewers:

- Prof. Dr. Dirk Timmermann (University of Rostock, Faculty of Computer Science and Electrical Engineering, Institute of Applied Microelectronics and Computer Engineering)
- Prof. Dr. Gero Mühl (University of Rostock, Faculty of Computer Science and Electrical Engineering, Institute of Computer Science, Research group Architecture of Application Systems)
- Prof. Dr. Martin Mauve (University of Düsseldorf, Computer Science Department, Group of Computer Networks)

Day of submission: 24 September 2018
Day of defense: 10 May 2019
Day of trial lecture: 27 May 2019

Abstract

Devices are becoming more powerful and are able to communicate with each other. These prerequisites have contributed to the development of the Internet of Things (IoT), in which various devices together form a large distributed system for, e.g., realizing the Smart Home vision. This trend continues as the fourth industrial revolution in industrial automation environments, into which the IoT has now found its way, often referred to as the Industrial Internet of Things (Industrial IoT, IIoT). In addition to the high scalability, accuracy, robustness, and performance, the real-time processing and communication capability plays a crucial role in realizing the future Smart Factory, as certain actions need to be completed on time to avoid damage.

This habilitation thesis explores design possibilities of networks and applications to realize reliable distributed systems that can meet these requirements. First, it examines network and application types for wired environments. A survey of real-time capable Ethernet solutions shows that there is currently no implementation that meets all requirements in the future Smart Factory. Therefore, an own development is presented, which takes up the weak points of existing solutions and realizes a real-time alternative with the aid of a decentralized peer-to-peer approach. Although the omission of a central entity leads to a high reliability and scalability of this solution, however, the performance and real-time capability compared to centrally controlled systems are limited. Subsequently, a real-time-capable system is presented, which uses a central SDN controller to compute and deploy scheduling and routing rules in networks, which are able to enforce the rules, for example, with the emerging time-sensitive networking technology. Several algorithms are presented, the former of which is suitable for the offline creation of scheduling and routing rules, but is limited in its scalability and can not react directly to changes at runtime. Therefore, additional algorithms are presented for the dynamic recalculation and execution of rules at runtime that scale better, but are currently limited to routing rules.

In the following, this thesis explores design options for reliable distributed systems in wireless environments. For wireless mesh networks, a management solution is introduced that acts as a middleware to increase the robustness of the mesh network and also enables cross-layer collaboration between different communication layers. The collaboration between mesh link layer and BitTorrent at application layer shows that the network and application performance can be increased significantly if both layers cooperate with each other. Finally, mobile opportunistic networks are investigated, in which the high mobility of the network participants imposes tight constraints on the convergence speed of algorithms. Two developed algorithms for distributed voting and people counting show how to realize distributed systems with high accuracy and scalability under these constraints. It is demonstrated that the systems achieve very good results in this respect for both synthetic and real movement patterns.

Kurzreferat

Geräte werden immer leistungsfähiger und sind zudem in der Lage, miteinander zu kommunizieren. Diese Voraussetzungen haben zu der Entwicklung des Internets der Dinge (Internet of Things, IoT) beigetragen, in dem diverse Geräte miteinander ein großes verteiltes System bilden und so z.B. die Smart-Home-Vision verwirklichen. Dieser Trend setzt sich als vierte industrielle Revolution in industriellen Automatisierungsumgebungen fort, in welche das IoT mittlerweile vorgedrungen ist und hier oft als industrielles Internet der Dinge (Industrial IoT, IIoT) bezeichnet wird. Neben der hohen Skalierbarkeit, Genauigkeit, Robustheit und Leistungsfähigkeit spielt hier die Echtzeitfähigkeit der Verarbeitung und Kommunikation eine entscheidende Rolle, um die zukünftige Smart Factory zu realisieren, da bestimmte Aktionen fristgerecht ausgeführt werden müssen, damit kein Schaden entsteht.

Diese Habilitionsschrift untersucht Entwurfsmöglichkeiten von Netzwerken und Anwendungen, um zuverlässige verteilte Systeme zu realisieren, die diese Anforderungen erfüllen können. Es werden zunächst Netzwerk- und Anwendungsmacharten für drahtgebundene Umgebungen untersucht. Eine Bestandsaufnahme von echtzeitfähigen Ethernet-Lösungen zeigt, dass es momentan keine Realisierung gibt, die alle Anforderungen in der zukünftigen Smart Factory erfüllt. Daher wird eine Eigenentwicklung vorgestellt, die die Schwachstellen bestehender Lösungen aufgreift und mit Hilfe eines dezentralen Peer-to-Peer-Ansatzes eine echtzeitfähige Alternativlösung realisiert. Obgleich der Verzicht auf eine zentrale Instanz zu einer hohen Ausfallsicherheit und Skalierbarkeit dieser Lösung führt, sind jedoch die Leistungsfähigkeit und Echtzeitfähigkeit gegenüber zentral gesteuerten Systemen eingeschränkt. Anschließend wird daher ein echtzeitfähiges System vorgestellt, das mit Hilfe eines zentralen SDN-Controllers berechnete Scheduling- und Routing-Regeln in Netzen ausbringt, die zum Beispiel durch die aufkommende Time-Sensitive-Networking-Technologie in der Lage sind, die Regeln durchzusetzen. Es werden verschiedene Algorithmen vorgestellt, von denen sich ersterer für die Offline-Erstellung von Scheduling- und Routing-Regeln eignet, allerdings in seiner Skalierbarkeit beschränkt ist und nicht direkt auf Änderungen zur Laufzeit reagieren kann. Daher werden weitere Algorithmen für die dynamische Neuberechnung und Ausbringung von Regeln zur Laufzeit vorgestellt, die besser skalieren, allerdings momentan auf Routing-Regeln beschränkt sind.

Nachfolgend widmet sich diese Arbeit Entwurfsmöglichkeiten von zuverlässigen verteilten Systemen in drahtlosen Umgebungen. Für drahtlose Mesh-Netzwerke wird eine Management-Lösung vorgestellt, die als Middleware agiert, um die Robustheit des Mesh-Netzes zu erhöhen und zudem die Cross-Layer-Kooperation zwischen unterschiedlichen Kommunikationsschichten zu ermöglichen. Anhand der Kooperation zwischen Mesh-Sicherungsschicht und BitTorrent auf Anwendungsschicht wird gezeigt, dass die Leistungsfähigkeit von Netzwerk und Anwendung deutlich gesteigert werden kann, wenn beide Schichten aufeinander Rücksicht nehmen. Schlussendlich werden mobile opportunistische Netze untersucht, in denen die hohe Mobilität der Netzteilnehmer der Konvergenzgeschwindigkeit von Algorithmen enge Grenzen auferlegt. Anhand von zwei entwickelten Algorithmen für verteilte Abstimmungen sowie Personenzählungen wird gezeigt, wie man dennoch verteilte Systeme mit hoher Genauigkeit und Skalierbarkeit unter diesen Randbedingungen realisieren kann. Es wird demonstriert, dass die Systeme diesbezüglich sehr gute Ergebnisse sowohl für synthetische als auch reale Bewegungsmuster erreichen.

Acknowledgments

At this point, I would like to take this opportunity to thank my family, who has always supported me with self-sacrifice in my now 30-year educational path. Without this support, the constitution of this research would not have been possible. My thanks also go to professor Dirk Timmermann and professor Gunnar Karlsson, who created the basis for the preparation of this research work by their technical support and my employment at the Institute of Applied Microelectronics and Data Technology of the University of Rostock and the Department of Network and Systems Engineering of the KTH Royal Institute of Technology, respectively. They helped me to master lows and difficulties, always motivated me and contributed significantly to the success of this research work. I would particularly like to thank Sylvia T. Kouyoumdjieva, György Dán, James Gross, and André Berger as well as Frank Golatowski, Martin Kasparick, Henning Puttnies, Eike Björn Schweißguth, Michael Rethfeldt, and Benjamin Beichler. In addition, I thank Henning Puttnies, Eike Björn Schweißguth, Michael Rethfeldt, and Benjamin Beichler, who have diligently read this thesis and all the colleagues and students involved in this work. Finally, I would like to say a big thank you to all employees of the Institute for Applied Microelectronics and Data Technology and the Department of Network and Systems Engineering, who made it possible for me to work in a friendly and nice atmosphere.

Contents

Contents IX

Acronyms XI

1 Introduction 1
1.1 Problem Definition and Outline of Contributions . . . 2

2 P2P-based Real-Time Ethernet 5
2.1 Real-Time and Real-Time Systems . . . 5
2.2 Real-Time Ethernet . . . 6
2.3 Suitability of P2P Networks for IoT and IIoT . . . 7
2.4 The HaRTKad System . . . 7
2.4.1 Synchronization . . . 7
2.4.2 Design Concept, Implementation, and Result Summary . . . 8
2.5 Summary and Outlook . . . 9

3 SDN-Supported Time-Sensitive Ethernet 11
3.1 SDN Basics . . . 11
3.2 SDN and Real-Time . . . 13
3.3 TSN Technology for Network Elements . . . 14
3.4 The SDN-Supported Real-Time Ethernet System . . . 17
3.4.1 Design Concept, Implementation, and Result Summary . . . 17
3.5 Routing and Scheduling as Optimization Problem at Design Time . . . 19
3.6 Algorithms for Dynamic Flow Migration at Runtime . . . 19
3.7 Summary and Outlook . . . 20

4 Management and Cross-Layer Solutions for Mesh Networks 21
4.1 WLAN Mesh Networks Basing on IEEE 802.11s . . . 21
4.2 Mesh Network Management . . . 23
4.3 Cross-Layer Collaboration between IEEE 802.11s and BT Networks . . . 24
4.4 Summary and Outlook . . . 25

5 Mobile Distributed Applications in Opportunistic Networks 27
- 5.1 Modeling Mobility 27
- 5.2 Distributed Voting 29
- 5.3 Distributed Crowd Counting 31
- 5.4 Summary and Outlook 32

6 Summary of Original Work 33

7 Conclusions and Future Work 45

Bibliography 51

A Survey on Real-Time Communication Via Ethernet in Industrial Automation Environments 59

B HaRTKad: A Hard Real-Time Kademlia Approach 77

C ILP-Based Joint Routing and Scheduling for Time-Triggered Networks 93

D Dynamic Flow Migration for Delay Constrained Traffic in Software-Defined Networks 115

E Integration of QoS Parameters from IEEE 802.11s WLAN Mesh Networks into Logical P2P Overlays 133

F DiVote: A Distributed Voting Protocol for Mobile Device-to-Device Communication 153

G UrbanCount: Mobile Crowd Counting in Urban Environments 175

Theses i

Teaching iii

Student Works v

Statement of Authorship ix

Acronyms

ALM	Airtime Link Metric
API	Application Programming Interface
BT	BitTorrent
CNC	Centralized Network Configurator
CUC	Centralized User Configurator
D2D	Device-to-Device
DEI	Drop Eligible Indicator
DHCP	Dynamic Host Configuration Protocol
DHT	Distributed Hash Table
DNS	Domain Name System
eMAC	express MAC
gPTP	generalized Precision Time Protocol
HaRTKad	Hard Real-Time Kademlia
HWMP	Hybrid Wireless Mesh Protocol
ICT	Inter-Contact Time
IE	Industrial Ethernet
IEEE	Institute of Electrical and Electronics Engineers
IET	Interspersing Express Traffic
IoT	Internet of Things
IIoT	Industrial Internet of Things
ILP	Integer Linear Programming
ISO	International Organization for Standardization
IP	Internet Protocol
JSON	Java Script Object Notation
LLDP	Link Layer Discovery Protocol
MAC	Media Access Control
MIB	Management Information Base
NTP	Network Time Protocol
OS	Operating System
OSI	Open Systems Interconnection
P2P	Peer-to-Peer
PCP	Priority Code Point
pMAC	preemptable MAC
PTP	Precision Time Protocol
RSSI	Received Signal Strength Indicator
RT	Real-Time
SDMA	Space Division Multiple Access
SDN	Software-Defined Networking
SNMP	Simple Network Management Protocol
SPoF	Single Point of Failure
TCI	Tag Control Info
TAS	Time Aware Shaper
TPID	Tag Protocol ID
TCP	Transport Control Protocol
TDMA	Time Division Multiple Access

TSN	Time-Sensitive Networking
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
WLAN	Wireless Local Area Network

1 Introduction

Computer systems are subject to a continuous advancement due to increasing processing capabilities of devices and network components as well as emerging communication technologies. As a result, today it is possible to compile computer systems consisting of a large number of devices connected by various communication networks. The number of devices that can potentially be networked is rising due to the development towards the Internet of Things (IoT). Recently, the IoT has even found its way into industrial environments. This is then referred to as Industrial Internet of Things (IIoT) and Industrial Internet, respectively [1, 2].

On the one hand and as opposed to central systems, distributed systems can potentially be realized in a more scalable and robust way. These characteristics are a direct consequence of the independence of devices. At the same time, distributed systems hide from the users and applications how each device contributes to the overall system. A distributed system should usually be available near-continuously even though some components may fail. On the other hand, the distributed system should look like a single system despite of the presence of heterogeneous devices and networks [3]:

> *A distributed system is a cluster of independent computers, which appear like a single coherent system to the users.*

Depending on its application, distributed systems might need to fulfill different requirements. Generally, distributed systems should be **scalable** while being comparably **accurate** like a central system. As mentioned above, a distributed system should be **robust** to leaving or failing devices and interferences implied by the communication medium and as well as be **high-performance**. In industrial environments, distributed systems moreover need to be not only robust but also **Real-Time (RT) capable**. If a distributed system meets the specific requirements demanded by its application it is further referred to as reliable distributed system.

The requirements, of which a reliable distributed system must meet those required by the particular application, are listed below.

- Scalability: The performance and accuracy of a system should not suffer from an increasing number of participants.
- Accuracy: Nodes should be able to compute accurate local estimates of global network parameters even though no central instance must exists, e.g., due to privacy reasons.
- Robustness: The system should be robust to failures such as misconfiguration, disturbances, or hardware failures and work on correctly at appearance of those. For providing error recovery capabilities for new networking technologies, management strategies are needed so that nodes can rejoin the network after such failures.

- High-performance: Protocols designed for reliable networks can benefit from exchanging information with other protocol layers to achieve high-performance communication even over unreliable network links.
- RT capability: Medium access in contention-based networks has to be timed to guarantee deterministic communication latencies since deadline misses may lead to disastrous consequences.

Therefore, this thesis proposes several network and application designs to realize reliable distributed systems in wired and wireless environments. Methodologically, the focus is first on wired networks with RT capability and afterwards shifts to wireless networks with static and finally to moving nodes to show a broad understanding of the topic by the author. The thesis orders own research results with regard to this methodology and thereby aims at an eased classification. By doing so, all chapters can be presented in a doctrinal way to be used for university courses.

1.1 Problem Definition and Outline of Contributions

The main objective of this thesis is to investigate how different kinds of distributed systems can be designed in such a way that they fulfill requirements demanded by their applications. To this end, the thesis is divided into four distinct parts.

The first two parts deal with middleware designs for achieving **RT capability** in wired Ethernet networks. In the first part, a decentralized design of **RT capable** Ethernet networks is introduced, which bases on the Distributed Hash Table (DHT)-based Peer-to-Peer (P2P) network Kad. The second part investigates how Software-Defined Networking (SDN) and Time-Sensitive Networking (TSN) technology can help to enable **RT** communication in industrial environments.

The second two parts address designs for **high-performance** and **accurate** operation of wireless mesh and opportunistic networks. The third part is dedicated to introducing management solutions into Institute of Electrical and Electronics Engineers (IEEE) 802.11s mesh networks to enable **robust** and **high-performance** applications in unreliable wireless environments. Finally, protocol designs for achieving **scalable** and **accurate** distributed applications in opportunistic networks with moving nodes is investigated in part four.

Following, the author lists in more detail the research question in each part and outlines the contributions.

Part I: P2P-based RT Ethernet

Part I (chapter 2) addresses the question if existing Industrial Ethernet (IE) approaches can keep in step with a rising numbers of devices to be connected and increasing amounts of data to be exchanged in RT in the future IIoT and IoT. It is investigated if purely software-based distributed approaches can enable RT communication over Ethernet and which cycle times can be achieved.

- **Paper A** gives a summary of different IE protocols for the RT data transmission via Ethernet in automation environments. Advantages and disadvantages of these protocols are analyzed with regard to their sustainability in terms of their RT capability, reliability, scalability, self-configuration of the network, and hardware requirements. Against the background of connected devices tremendously growing in number and computational power in the prospective IIoT, consequences for future developments are drawn.
- **Paper B** presents an approach to realize a hard RT network for automation scenarios using P2P technology. Kad is an implementation variant of the structured decentralized P2P protocol Kademlia and has been chosen as base for the realization and is extended by a Time Division Multiple Access (TDMA)-based mechanism. The performance is evaluated by means of a prototype implementation.

Part II: SDN-Supported Time-Sensitive Ethernet

Part II (chapter 3) is dedicated to the question, which is the method of choice for routing and scheduling of RT networks if complex distributed application constraints are to be considered and how SDN and TSN technology can help to enforce the desired network behavior. Further, the question is investigated how algorithms that scale with increasing scheduling and routing problem sizes have to be designed. Especially, the focus is put on networks with RT requirement that change dynamically at runtime.

- In **paper C**, a novel Integer Linear Programming (ILP) formulation is presented that can be used to jointly solve the routing and scheduling problem for RT Ethernet networks. Using this formulation, it is possible to solve various scheduling problems that are infeasible when using a fixed shortest path routing with separate scheduling. Compared to a fixed load balanced routing with separate scheduling, schedules computed with the formulation can offer lower communication latencies.
- In **paper D**, the problem of dynamic flow migration, i.e., a routing reconfiguration is considered and a polynomial time algorithm is proposed that can find a solution if direct flow migration is feasible. Furthermore, an algorithm for computing both direct and indirect flow migration is proposed and its correctness is proven. Numerical results obtained on a FatTree network topology are presented.

Part III: Management and Cross-Layer Solutions for Mesh Networks

Part III (chapter 4) explores the question of how to improve the network management of the mesh standard IEEE 802.11s and how a network management solution should be designed in order to improve the robustness of such networks. It is then examined how cross-layer cooperation between the link layer of wireless mesh networks and overlying layers can help to improve performance.

- **Paper E** presents a cross-layer approach based on 802.11s and BitTorrent (BT), that optimizes application layer peer selection by considering the mesh standard's routing metric Airtime Link Metric (ALM) provided by the developed network management solution. The solution is implemented and evaluated in a real-world test bed. Results show that average download time can be reduced by up to 20 % already in small network setups.

Part IV: Mobile Distributed Applications in Opportunistic Networks

Part IV (chapter 5) investigates fundamental limits and possibilities of dynamic opportunistic networks of mobile nodes under various conditions in order to evaluate their scalability and accuracy. As a prerequisite, it is assumed that there are tasks that the mobile devices can complete cooperatively such as collecting and aggregating data. After having processed this data, the nodes can distribute the results to other interested nodes. This procedure describes a collaborative, participatory, and cooperative sensing and is often referred to as mobile cloud computing. The challenge is to reliably carry out the distributed computation under varying operating conditions with moving nodes and high churn. Distributed voting and counting serve as the two lead scenarios.

- In **paper F**, DiVote is presented, which is a distributed voting protocol in the context of urban polling that is suitable for environments, in which nodes exhibit high mobility. It relies on Device-to-Device (D2D) communication to exchange voting information. The dynamism due to mobility imposes tight constraints on the speed of the algorithm, with which its computed estimates converge to the global result. Hence, DiVote immediately updates the local estimate. Simulation results obtained when applying DiVote to realistic pedestrian mobility traces show that even in sparse scenarios local estimates quickly converge to the global result.
- **Paper G** describes UrbanCount, a fully distributed protocol for crowd counting via D2D communication, which is suitable for operation in urban environments with high node mobility and churn. Each node collects crowd size estimates from other participants in the system whenever in direct communication range, and immediately integrates these estimates into a local estimate. The proposed protocol is evaluated via extensive trace-driven simulations of synthetic and realistic mobility models, and two main questions are answered: Under which circumstances is distributed counting possible, and how accurate can it be? It is shown that for sufficient densities distributed crowd counting always produces precise estimates regardless of the area in which mobility occurs. Furthermore, UrbanCount provides a scalable solution for crowd counting for both indoor and outdoor scenarios.

2 P2P-based Real-Time Ethernet

The IoT has already found its way into smart home and building environments. Devices like refrigerators can be accessed by their Internet Protocol (IP) addresses and can seamlessly exchange information with other devices. However, this is not the only scenario, in which embedded systems are networked to integrate them into one system. In the context of industrial automation, the fourth industrial revolution has been predicted to happen leading to an intelligent manufacturing. If RT capable devices are networked based on IP, the trend can generally be referred to as IIoT. In Germany, the specific term “Industrie 4.0” has been coined for this development [2]. Moreover, the US-American company General Electric initiated a comprehensive research initiative called “Industrial Internet” and predicts an increasing number of IP-connected devices even in industrial environments [1].

This chapter deals with a distributed approach for networking RT capable devices via Ethernet in automation environments. It differs from established RT Ethernet solutions such as Ethercat, TTEthernet, Profinet IO/IRT, SERCOS III, and CC-Link IE in that it uses the P2P protocol Kad rather than a central master node to time the medium access of each device. An exhaustive comparison of existing solutions is provided in **paper A** included in this thesis. **Paper A** motivates the P2P-based RT Ethernet system called Hard Real-Time Kademlia (HaRTKad), which is then described in **paper B**. In what follows, the author provides basics and information that serve as an introduction and summary to the HaRTKad system. Finally, the HaRTKad system is classified with regard to ongoing research activities in the field of SDN and TSN.

2.1 Real-Time and Real-Time Systems

First of all, it is necessary to define the term RT. The correctness of a system, which processes data in RT, is not only depending on the computed results but also on the time, in which the results have been determined [4].

A standardized definition of a RT system is given in [5]:

> *If the operation of a system, in which programs for data processing are operational at any time, is such that results are available within a defined time interval, the system is denoted as RT system. Depending on the use case, results can be available after a temporally random distribution or at predetermined points in time.*

Generally speaking, a RT system consists of an external and an internal system. The external system provides the internal system with time constraints concerning the processing of tasks. The internal system has to consider these constraints. However, one has to differ between soft and hard RT depending on the strictness of the time constraints. Figure 2.1 depicts a comparison of benefit and damage, respectively, for a task with soft and hard RT requirements. If a task has soft RT requirements, no damage occurs if

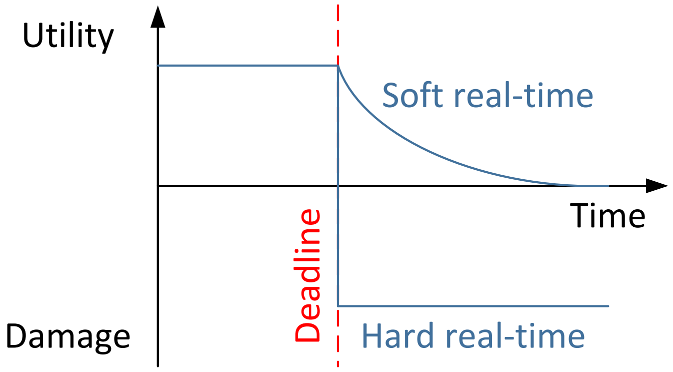


Figure 2.1: Comparison of utility and damage, respectively, for a task with hard and soft RT requirements.

the deadline for processing the task is violated. However, the utility of it decreases the longer it takes to complete the task after the deadline. On the contrary, a task with hard RT requirements has to meet the deadline as otherwise the utility is zero and damage occurs. Formally speaking, a task, which starts at point in time r and takes time Δe to be completed, has to be finished at point in time d. This condition is also referred to as timeliness and is apparent from equation 1:

$$A \equiv r + \Delta e \leq d \tag{1}$$

To ensure hard RT, the time condition A has to be valid given constraint B (cf. equation 2). In direct relation to the respective task, B stands for the fact that in the period from r to d neither technical failures occur nor more important tasks are to be accomplished.

$$P(A|B) = 1 \tag{2}$$

Distributed RT systems consist of several single systems, which have to communicate with each other, as tasks are distributed among them. To ensure the RT behavior of the overall system, each single system as well as the communication between them has to be RT capable. Whereas the RT capability can be achieved by choosing an appropriate embedded system running a RT Operating System (OS), further measures has to be taken to guarantee RT communication. In what follows, Ethernet is selected as communication medium and measures for realizing RT Ethernet are detailed.

2.2 Real-Time Ethernet

Standard Ethernet nowadays uses switches to avoid collisions. However, due to buffering in the switches and potential buffer overflow in case of traffic peaks, the timeliness of data delivery cannot be guaranteed and data can be discarded. To ensure timeliness and to avoid packet loss, established RT Ethernet solutions use TDMA as medium access method. Usually, a master component keeps track of slave nodes, which want to communicate cyclically. It then assigns time slots to these nodes so that a node can exclusively access the Ethernet medium during its time slot. For a brief description and comparison of state-of-the-art RT Ethernet systems, the interested reader is referred to **paper A** included in this thesis.

2.3 Suitability of P2P Networks for IoT and IIoT

In the work of Danielis et al. [6], the author investigates P2P overlay networks for Ethernet-based access networks. P2P networks show advantages compared to centralized approaches concerning reliability, self-organization of the network, and scalability with regard to the number of network nodes. The analyzed overlay protocols are located at the application layer and therefore do not require special hardware or modifications at lower layers, i.e., they can be implemented on top of the standard protocol stack (Ethernet, IP, User Datagram Protocol (UDP)/Transport Control Protocol (TCP)). In this regard, the author identified the DHT-based P2P protocol Kad [7, 8] to be particularly suited for implementing distributed applications. In order to ensure a deterministic search for devices and data in a distributed environment, the author developed an algorithm to dynamically adapt a search parameter in the Kad network at runtime [9].

Based on the optimized Kad network, the author realized distributed network applications by means of the networking of access networks' access nodes. Thereby, the advantages of Kad (realibility, self-organiziation of the network, and scalability) could be transferred to these applications. Access nodes represent active network components of access networks, which aggregrate the data traffic of connected clients, forward it to the core network, and provide Internet access to clients. As a result, the author developed concepts and prototypes for a distributed Domain Name System (DNS), for the reliable distributed storage of Internet access data, and for the distributed computation of statistics [10, 6, 11]. These works hence show that embedded devices have become sufficiently capable to execute a P2P protocol such as Kad at the application layer. Consequently, they are of interest for devices in the IoT and prospectively in the IIoT.

2.4 The HaRTKad System

However, the Kad protocol is not RT capable without further adaptations. It has hence to be modified for enabling the RT processing on devices as well as a timed access to the Ethernet medium. The resulting RT capable Kad networks is called HaRTKad [12]. The main idea behind HaRTKad is to correlate the hash values assigned to each Kad node in the network to time slots.

2.4.1 Synchronization

To lay the foundation for the HaRTKad nodes to be able to adhere to their time slots, the author developed a decentralized time synchronization algorithm in collaborative works [13, 14] rather than deploying a centralized synchronization protocol such as

Precision Time Protocol (PTP) or Network Time Protocol (NTP). The algorithm organizes devices into a logical tree structure, in which parent nodes synchronize a variable number of child nodes. Based on the established common time base, a time slot is assigned to each device taking into account its hash value, in which it can send data to other devices.

2.4.2 Design Concept, Implementation, and Result Summary

To enable the RT processing of the HaRTKad protocol by executing it as RT task on embedded systems, a RT OS such as FreeRTOS finds application (see Figure 2.2). Devices communicate by means of standard Ethernet whereby the Ethernet driver and the UDP/IP communication stack are part of FreeRTOS. As communication stack, the author selected the lwIP stack (lightweight IP) [15], which has been developed for the application on embedded systems. During a time slot controlled by the HaRTKad protocol, a device can perform a complete P2P interaction (i.e., request the destination node, process data, and receive the response) with other devices. Note that solely the hosts executing the HaRTKad protocol adhere to time slots by implementing a time-controlled queue. Experimental result show that the HaRTKad protocol is RT capable and can hence serve as middleware for the execution of RT applications.

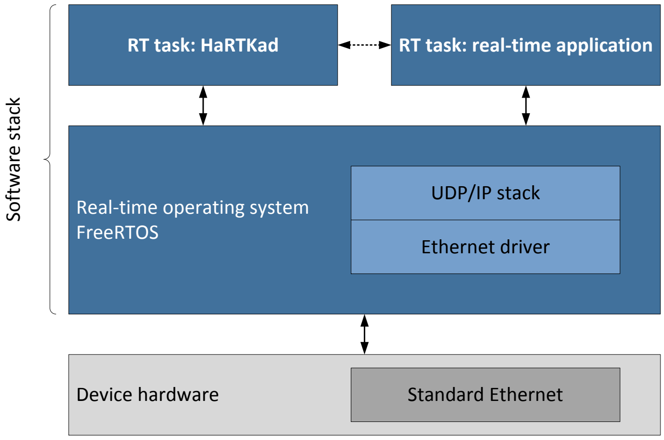


Figure 2.2: Software stack of the HaRTKad protocol on top of device hardware.

An example network consisting of four devices with HaRTKad functionality is apparent from Figure 2.3. Each device can communicate during an exclusive time slot determined by its hash value. Such a network consisting of a few Zedboards with 667 MHz ARM processor connected by a 1 Gbit/s Ethernet switch has been setup as proof of concept and cycle times of 1 ms could be reached. All devices run an RT application that cyclically sends one UDP datagram with maximum transmission unit size of 1514 byte to another device. A further analysis shows that without further optimizations a cycle time of 1 ms with 85 devices is possible. A detailed description of the HaRTKad system is given in

paper B included in this thesis.

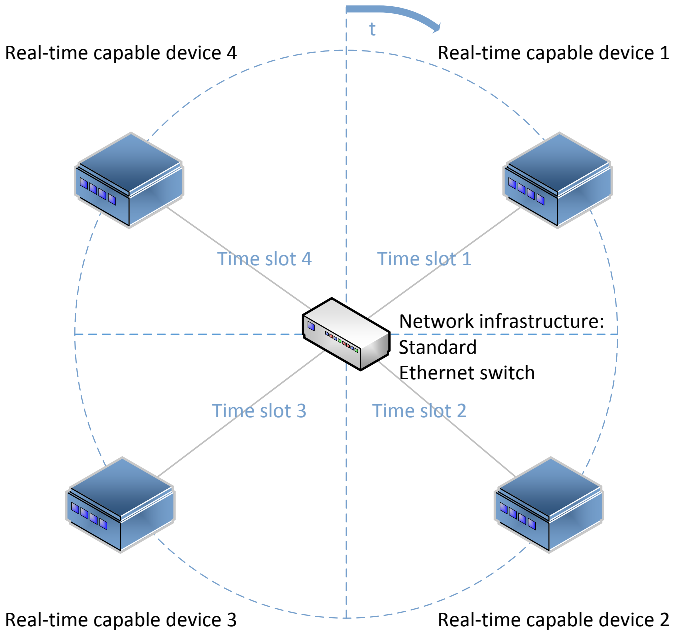

t
Real-time capable device 4
Real-time capable device 1
Time slot 4
Time slot 1
Network infrastructure:
Standard
Ethernet switch
Time slot 3
Time slot 2
Real-time capable device 3
Real-time capable device 2

Figure 2.3: An example network consisting of four devices with HaRTKad functionality.

2.5 Summary and Outlook

HaRTKad is a purely software-based distributed approach to RT communication over Ethernet with cycle times of 1 ms. Due to its distributed nature, it can dispense with a central instance and therefore shows the advantages of potential scalability and reliability. Note that a detailed analysis of the scalability regarding the achievable number of devices, which are able to communicate under consideration of their RT requirements, is subject to future work.

However, the advantages of a distributed approach such as HaRTKad have to be traded-off against the limitations regarding its performance as pointed out in [16]. First of all, a possibly low network utilization in cases, in which nodes are clustered in a small area of the available logical address space, can occur. This may result in a high number of potential time slots, of which only a small fraction is actually be used. Secondly, hash collisions can appear if two or more nodes have the same hash value leading to the same slot time to be used by them. Thirdly, HaRTKad cannot prioritize certain time-sensitive data traffic in front of best-effort traffic by default.

The work in [16] proposes solutions to mitigate these problems, however the achievable performance still remains limited by the lack of a central instance, which has the overview over the whole network. Such an instance like an SDN controller can leverage the network overview to compute a schedule for assigning concurrent time slot to nodes, whose communication takes place over non-overlapping network links. While this approach does not violate RT constraints, it can potentially increase the network utilization and at the same time meet the RT requirements of an increasing number of nodes and therefore motivates the use of SDN.

Moreover, as mentioned above only HaRTKad nodes adhere to time slots while Ethernet switches do not and time-sensitive traffic cannot be prioritized. Switches are neither time-controlled nor configurable and hence cannot be influenced concerning their switching behavior. This makes HaRTKad more suitable for networks with only RT traffic of small to medium size, in which the delay resulting from traversed switches can be taken into account as an upper bound when calculating the worst-case communication delay. On the contrary, the new IEEE standard technology TSN introduces new features into switches, which enable them to adhere to time slots and self-configured routing rules as well as to prioritize RT traffic in front of best-effort traffic [17]. In combination with an SDN controller that calculates scheduling and routing rules, TSN-enabled switches allow for combined RT and best-effort networks of larger scale compared to HaRTKad at the expanse of introducing a central instance, more complex technology into switches, and higher configuration effort. A more detailed analysis of the benefits achievable with SDN and TSN technology is given in the following chapter 3.

3 SDN-Supported Time-Sensitive Ethernet

RT capable communication systems in industrial environments historically and still today often base on fieldbuses. However, RT capable Ethernet networks are increasingly replacing traditional fieldbuses. Recently, Ethernet tailored to the requirements in industrial environments has overtaken fieldbuses in terms of the number of newly installed nodes in factory automation [18]. This development results from the fact that systems that base on Ethernet can be integrated easily into existing IT infrastructures and allow for a consistent networking. Like fieldbuses, established RT Ethernet solution are mostly proprietary and achieve their RT capability by means of solution-tailored modifications at the link layer (cf. **paper A** included in this thesis for an exhaustive comparison).

The SDN approach represents a suitable technology for enabling the development of new RT Ethernet based solutions by means of purely software-based measures rather than link layer modifications. Due to the programmability and central overview of the SDN controller, flexible networks with complex routing strategies and extended user-defined functions can be realized as opposed to traditional networks. However, it depends on the applied algorithms whether RT capability is introduced into the network. Algorithms that compute schedules and routes (cf. **paper C** included in this thesis) or reserve routes (cf. **paper D** included in this thesis) for time-sensitive applications are suitable for enabling RT behavior of a network. Further, network elements have to be capable of adhering to the computed schedules, bandwidth reservations, and routes to enforce the RT behavior. In this regard, technologies like TSN provide the necessary functionality to enable time-sensitive network elements [17, 19]. Following sections detail the working principle of SDN and introduce RT related aspects.

3.1 SDN Basics

Traditional network elements comprise own control logic, which analyzes incoming packets and decides about their processing. The control logic is integrated into the respective network element and connected to its hardware, which executes the envisaged processing steps. Typically, these steps consist in the directed packet forwarding with or without modification of specific header fields or the discarding of packets. The control logic distributed in the network is also referred to as control plane whereas the executing hardware is denoted as forwarding or data plane [20].

For changing the network configuration, traditionally there is no central controller but many single devices have to be configured individually. To facilitate the configuration, management tools are offered by network suppliers, which offer central configuration possibilities. However, due to different protocols and interfaces these tools do not offer comprehensive compatibility with hardware of different manufacturers [20].

The SDN paradigm is supposed to overcome existing problems and offers new possibilities to reconfigure networks. The control plane is separated from single network elements and moved to a central controller. The controller is implemented in software in order to ensure adaptability to specific requirements [21].

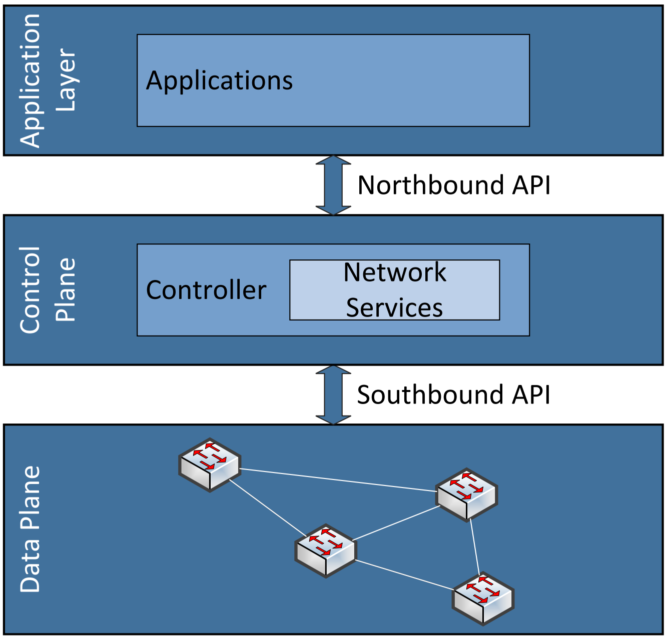


Figure 3.1: Overview of the SDN network structure.

Figure 3.1 depicts the SDN network structure. The controller communicates with elements of the data plane by means of a standardized interface, which is also referred to as southbound Application Programming Interface (API). OpenFlow is one well-known and widely used southbound API developed by Nick McKeown et al [22]. In what follows, network elements such as Ethernet switches are assumed to be OpenFlow-capable and are further referred to as OpenFlow switches. Since the elements of the data plane contain no own control logic, they do not decide about the processing of packets on their own. Instead, they use the southbound API to request the correct processing of incoming packets from the controller. To reduce the arising configuration traffic between network elements and controller, the controller can install rules in the single data plane elements, which determine the processing of specific packets. Network elements then solely have to issue requests to the controller if for an incoming packets no matching rule exists [21, 23]. The rule system of OpenFlow, for instance, works with so-called flows and actions. The term flow thereby refers to a certain communication relation in the network. Each rule consists of a flow definition and no to several assigned actions. The flow definition determines, for which packets a rule applies, and supports header fields such as source and destination Media Access Control (MAC) and IP addresses as well TCP and UDP ports. Actions comprise, among other things, the forwarding of a packet on a specific switch port, the modification of a packet's header fields, or the discarding of the packet.

In addition to the southbound API, the utilization of an optional northbound API is possible to realize the communication between controller and applications, which use the network. For the realization of the northbound API, an own protocol or REST-based protocol and already existing data exchange formats like Java Script Object Notation (JSON) can be used [24]. Via this interface, applications can for instance inform the controller about their requirements such as RT constraints in particular or quality-of-service demands in general.

3.2 SDN and Real-Time

By leveraging the possibilities offered by SDN technology, an efficient RT communication system can be realized for arbitrary topologies. In order to guarantee the RT capability of the communication, the SDN controller has to control the medium access. Therefore, all OpenFlow switches connect to an SDN controller upon network start-up and subsequently the controller discovers the network topology. In doing so, all connections between OpenFlow switches are discovered. Then, the controller searches for connected end devices, i.e., hosts. Finally, the controller has to be informed about the requirements of the end devices, which communicate with each other. This information consists of flows, the data volume to be exchanged, and the latency which is allowed at maximum.

By means of the topology data base and communication requirements, the controller can plan the utilization of each link in the network to allow exclusive access to it by a specific flow. The topology information can be used to allow for flows that communicate at the same time provided that they do not use the same links (Space Division Multiple Access (SDMA)). This represents the main advantage over the distributed HaRTKad approach and leads to a potentially increased network utilization. As mentioned before, HaRTKad cannot allow simultaneous communication by default without being at risk of violating RT requirement due to the lack of a central instance. If switches in the networks are OpenFlow-capable, they can be instructed by the controller which routes and involved links have to be reserved for which flows. An easy way to realize RT is to fully reserve routes for a flow. On the one hand and in contrast to HaRTKad, the end devices do not have to be synchronized for this approach. However, only very few flows can then communicate as a link is only available for a single flow, comparable to a connection-oriented approach.

One approach in addition to SDMA to further increase the network utilization while meeting RT requirements is to reserve specific bandwidth for flows, i.e., to accommodate more than one flow per link and to apply queuing management on switches in order to prioritize flows [25]. However, this requires switches to be capable of programmable queuing management mechanisms, e.g., offered by TSN technology [17], which is detailed in the subsequent section. On the one hand, network components still do not need to be synchronized, which simplifies their functionality. On the other hand, it remains unclear how to restrict the communication jitter without timing the communication.

Another approach is to determine *when* which flow is allowed to use which link. This flow-specific time slot method (TDMA) allows the assignment of several time slots per connected end device, in which it may send flow-specific data. Typically, a device in automation environments does not need medium access the entire time but solely for cyclically sending process data of some kBytes in size. However, to enable devices to adhere to their time slots they need to be synchronized and, due to clock drift, even be re-synchronized periodically [26, 27]. On the one hand, this causes traffic overhead in the network but on the other hand, enables an efficient scheduling of RT traffic with the ability to restrict the communication jitter. However, the complexity of solving such a routing/scheduling problem is NP-complete. While the complexity is already high when computing the routes and schedules offline before the network operation begins, it becomes even higher when recomputing routes and complete schedules at runtime to reflect changes in time. Even for small networks, dynamic changes are hardly possible. As mentioned above, to allow network components to adhere to their time slots, their transmission times have to be programmable, which is not possible with OpenFlow but is for instance enabled by TSN technology [17].

3.3 TSN Technology for Network Elements

The TSN task group defines functionality of network components for allowing the deterministic data transmission in Ethernet networks [17]. The TSN standard consists of several sub-standards, of which the ones that are of relevance for this thesis will be mentioned in the following.

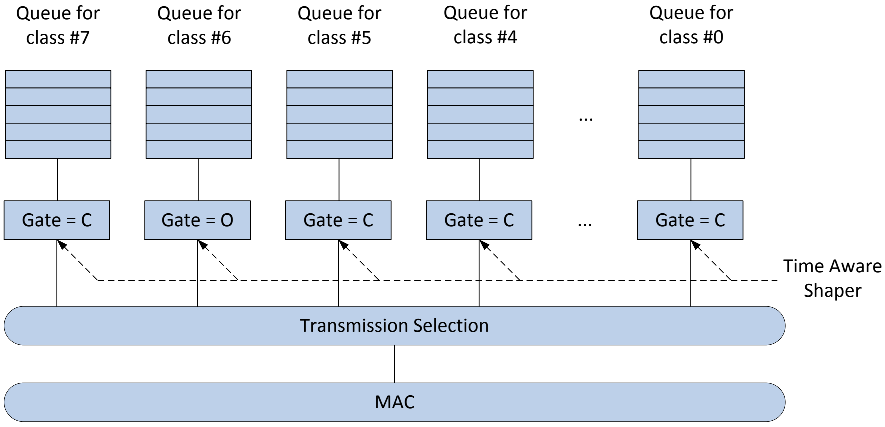


Figure 3.2: Stream classes defined in IEEE 802.1Q [28].

For the time synchronization of TSN devices, the IEEE 802.1ASrev sub-standard defines the generalized Precision Time Protocol (gPTP) as a generalization of the PTP. If switches support gPTP, they are referred to as Time Aware Bridges.

In order to ensure time-controlled communication, a component called Time Aware Shaper (TAS) defined in IEEE 802.11Qbv is implemented on TSN-capable switches to enforce that configured RT constraints for specific flows are followed. IEEE 802.1Q defines eight stream classes for Ethernet frames and according priorities (see Figure 3.2). On Time-Aware Bridges, for each class a queue and a related transmission gate are available. These gates can be opened (O) or closed (C) by the TAS according to a time-based control list (see Figure 3.3). Thereby, certain output queues of a switch can be selected for transmission and sending in time slots can be realized.

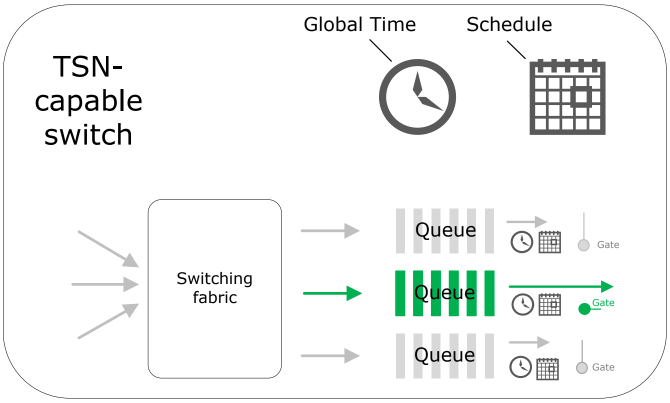


Figure 3.3: Operation scheduling by means of transmission gates controlled by the TAS [29].

The TSN technology further comprises the IEEE 802.1Qbu frame preemption sub-standard, which bases on the Ethernet sub-standard 802.3br [30]. It allows for the prioritization of RT traffic in front of other traffic like best-effort traffic by means of the introduction of express frames. As part of the Ethernet sub-standard Interspersing Express Traffic (IET), for each port of an Ethernet switch two MAC interfaces are specified. One interface serves as the basis for the transmission of express frames while the other is designated to the transmission of standard frames. These interfaces are called express MAC (eMAC) and preemptable MAC (pMAC) in the TSN sub-standard and are introduced into the MAC of TSN switches (see Figure 3.4). The transmission of standard frames can be preempted when an express frame arrives [31]. This results in a possible fragmentation of preemptable frames. Due to the fragmentation, the guard band, that stops standard frames from being transmitted before an express frame is scheduled, is reduced, which leads to a more efficient use of the network bandwidth. The fragments of preempted frames have to be defragmented at each network node as IEEE 802.1Qbu sub-standard does not support end-to-end fragmentation.

In order to enable the deployment of TSN sub-standards, among other things, the Q tag

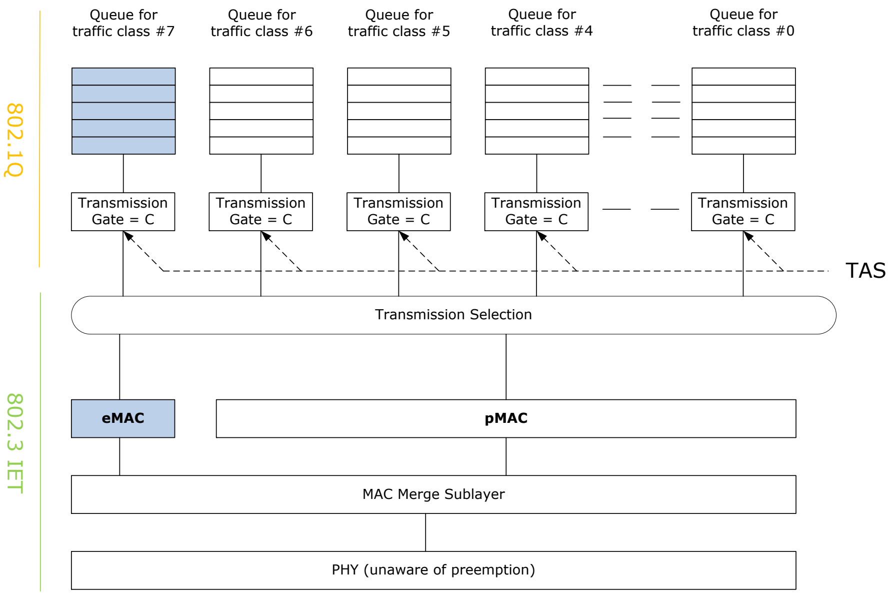


Figure 3.4: Introduction of eMAC and pMAC interfaces to enable TSN frame preemption [28].

is required to be present in Ethernet frames in order to be used by TSN-capable devices [32]. This tag is located between source MAC address and type field in the header of an Ethernet frame (see Figure 3.5). The field has a length of four bytes and contains two bytes for the Tag Protocol ID (TPID). It is set to the fixed value 0x8100 to signal that the frame contains information related to the 802.1Q standards [33]. The following two bytes comprise the Tag Control Info (TCI). It consists of three bits Priority Code Point (PCP). They can be used for assigning priorities to frames, which is of relevance for the frame preemption and TAS. Subsequently, the Drop Eligible Indicator (DEI) flag signals if data is to be discarded in case of delay. The final 12 bits contain a Virtual Local Area Network (VLAN) identifier.

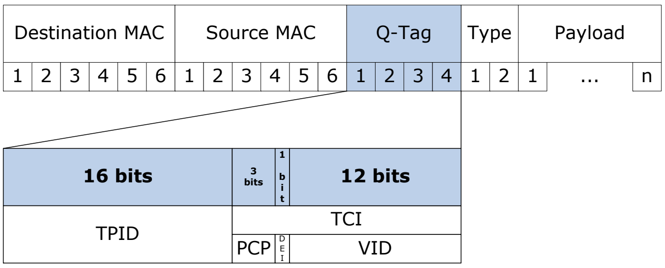


Figure 3.5: Structure of an Ethernet frame with Q tag [33].

Summarized, while TSN technology enables the configuration of Ethernet network components in such a way as to enforce RT behavior, the algorithms for calculating respective rules are out of scope. The proposed TSN architecture according to [17] to put into practice a RT capable Ethernet network is depicted in Figure 3.6. So-called flow requests can be issued between Centralized User Configurator (CUC) and Centralized Network Configurator (CNC) or CUC and end points. These requests contain identities and quality-of-service parameters such as bandwidth, latency, and topology to establish a TSN connection. Parameters are saved at the CUC. The CNC collects information with the help of the CUC to obtain detailed knowledge about topology and network parameters as well as network elements. Subsequently, it can compute a schedule as well as necessary bandwidths reservations, and routes and then deploy respective rules on the TSN-capable devices in the network. Compared to SDN, CUC and CNC are similar to a SDN controller where user-defined algorithms run, which calculate respective rules to achieve the RT behavior of the network. By means of these rules, the TAS controls the transmission gates of the queues on the switches.

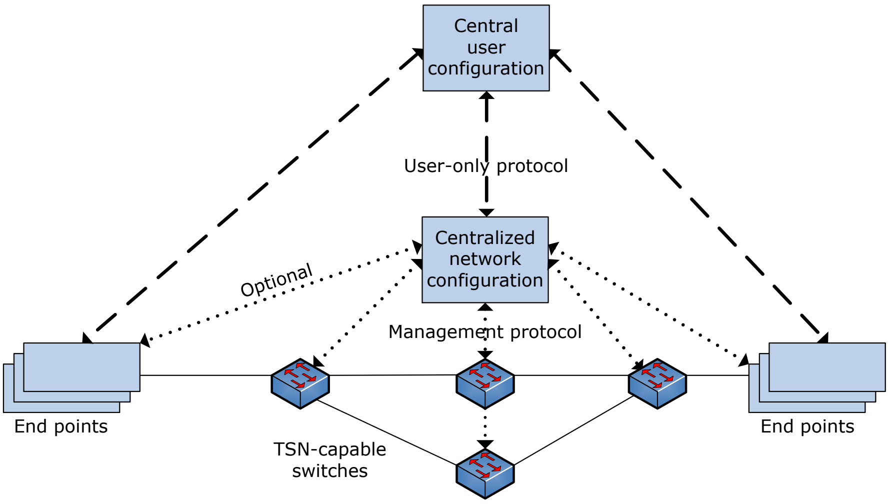


Figure 3.6: Proposed TSN architecture according to [17].

3.4 The SDN-Supported Real-Time Ethernet System

The author first leveraged the possibilities offered by SDN in a collaborative work to realize a SDN-supported time-sensitive Ethernet system in a practical test bed. The test bed consists of an OpenFlow-capable HP switch and eight ZedBoards as end points running FreeRTOS with the lwIP network stack [34].

3.4.1 Design Concept, Implementation, and Result Summary

The MAC of the developed system bases on the TDMA approach and is enhanced by taking into account topology information and controlling routes by means of the SDN controller. Thereby, multiple devices can send data over physically separated links simultaneously (SDMA). This combined TDMA/SDMA approach enables on the one hand hard RT communication. On the other hand, it enables the efficient utilization of network resources for a high number of network participants.

Before the RT communication in the network can start, the system runs through startup steps in order to configure the network properly. First, the 1 Gbit/s OpenFlow HP 2920 data center switch connects to the SDN controller, which is realized with the POX framework [35]. It discovers the topology with slightly modified POX discovery modules relying on the Link Layer Discovery Protocol (LLDP). Subsequently, the controller collects the information on the communication requirements of each end point, which include

- the time interval, with which an application creates data packets to be sent,
- the assigned time slot length,
- and the maximum allowed latency of the communication.

With the help of the topology information and communication requirements, the controller computes a route and a send schedule for each flow in the network. It uses a heuristic algorithm, which creates the schedule taking into account different time slot lengths and cycle times per flow. A modified version of Dijkstra's single source shortest path algorithm is executed, which is aware of time slots, to search for a possible route. Thereby, the algorithm allows for a conflict-free communication since at no point in time two flows are scheduled for transmission via the same link. The controller then installs routing rules into the OpenFlow switch via the OpenFlow southbound API and communicates the send schedules to the end points via the northbound API using a custom JSON format. Observe that solely the ZedBoard end points follow a send schedule with a time resolution of 10 μs while switches are not time-controlled. The used OpenFlow switch simply forwards incoming packets depending on configured OpenFlow rules, which it processes in hardware. Before the RT communication finally starts, the end points are synchronized.

The achieved latencies in the test bed are below 1 ms and can be expected to be far below 10 ms even with long multi-hop routes. The results confirm that the proposed system achieves comparable cycle times and latencies like existing RT Ethernet solutions but has superior scalability and uses standard Ethernet hardware. However, the heuristic algorithm solely finds a feasible solution to the joint routing/scheduling problem while optimal solutions might be able to solve more complex problem instances. Hence, in the next section the author gives an introduction to formulating the NP-complete joint routing/scheduling problem as optimization problem.

3.5 Routing and Scheduling as Optimization Problem at Design Time

In **paper C**, which is included in this thesis, the idea from [34] is extended and formulated as ILP optimization problem for joint routing/scheduling. The formulation bases on an objective function, which defines the sum of all flows latencies as optimization target, and constraints such as resource and application constraints, which have to be considered in the optimization. Different problem instances are fed into the ILP solver Gurobi, which generates optimal solutions in the form of routes and schedules for both end points and switches. As both switches and end points are expected to be able to adhere to time slots, they need to be synchronized before network operation. Test cases with up to 52 flows comprise three topologies (ring, 2 partial meshes) consisting of 12 switches and 12 host, which could be realized with TSN-capable Ethernet switches. As opposed to existing approaches, the joint routing/scheduling algorithm solves problems which are not solvable with shortest-path routing and separate scheduling. In the worst case, solely 33.3% of the traffic patterns that were schedulable by joint routing/scheduling could be successfully scheduled by shortest-path routing and separate scheduling. Compared to load-balanced routing and separate scheduling, the approach further achieves lower communication latencies for flows. Actually, load-balanced routing and separate scheduling shows up to 61.8% higher flow latencies. However, the proposed solution is still determined offline before network operation. To be able to accommodate additional flows at runtime, the following section describes algorithms which are able to operate online.

3.6 Algorithms for Dynamic Flow Migration at Runtime

The author addresses the problem of dynamic flow migration, i.e., the the dynamic routing reconfiguration of flows at runtime in **paper D** included in this thesis. As opposed to the previous section, solely the routing of flows is controlled and no time slots are calculated. The proposed algorithms are executed by an SDN controller to determine migration sequences. In these sequences, one path at a time is migrated with an elementary flow migration until the final network configuration is obtained such that one additional flow can be integrated. Thereby, path combinations are only valid if their edges are disjoint. An elementary flow migration applies atomic forwarding table updates to one switch after the other from the destination to the source of the path to be migrated. This way, the flow migration does not interrupt ongoing traffic and switches as well as end points do not need to be synchronized as opposed to related works. The switches solely need to support a protocol such as OpenFlow to be able to follow routing rules computed by the flow migration algorithms.

One of the proposed algorithm finds direct flow migration sequences, in which any path is migrated at most once. It has polynomial runtime but does not find all solutions for given problem instances. The second algorithm identifies direct and indirect flow migration sequences. It thus provides the full solution set for given problem instances but at the expense of increased computational complexity. Numerical results reveal that for a FatTree topology that is moderately loaded with flows, in 24% to 30% flow migration is necessary. In approximately 60% of these cases, the direct flow migration can solve the problem. In the other cases, the more complex algorithm needs to be executed, which shows high ratios of long computation times for loaded topologies. Summarized, **Paper D** extends recent works by proposing the polynomial time algorithm for deciding whether direct flow migration is feasible, and by evaluating the necessity and complexity of indirect FM.

As part of future work, the author of this thesis plans to evaluate the algorithms in a simulation environment that allows deploying atomic forwarding table updates and will consider other topologies as well as compare the algorithms to existing solutions. Furthermore, the author plans to relax the requirement of edge-disjointness and consider weighted graphs to allow for expressing finer-grained delay constraints.

3.7 Summary and Outlook

As shown in [34] and **paper C**, algorithms that compute routes and schedules for RT flows are able to guarantee low latencies in the order of some millisecond. One precondition for these algorithms to be able to ensure latencies in this order is the assumption that they are executed on a central instance like an SDN controller, which has the overview of the network topology and knows the RT requirements of involved communication partners. An ILP solver could optimally solve medium-size routing and scheduling problems (24 network nodes, $\leq$ 52 flows) in computation times of below 22 min, which is tolerable if routes and schedules can be computed offline before the network operation starts. However, if the algorithms often need to react to dynamism in the network at runtime even medium-size problem instances become an issue in terms of computation times. The approximation algorithm for direct flow migration presented in **paper D** shows polynomial runtime and can help to reduce computation times but is currently only able to reserve routes for flows.

Consequently, the algorithms are supposed to be extended for computing schedules as well. They will moreover be developed further to be able to achieve RT capability in wireless environments. Since the wireless medium is less reliable than wired Ethernet, ensuring RT communication is complicated even in the presence of TDMA mechanisms. Challenges to be faced when establishing reliable communication in wireless environment are detailed in the following Section 4. Moreover, the practical deployment of the algorithms in a network with TSN switches is subject to future work. The performance and RT capability of TSN technology in practical use is already researched in the 3-year ITEA3 project "OPTIMUM - OPTimised Industrial IoT and Distributed Control Platform for Manufacturing and Material Handling" supervised by the author of this thesis.

4 Management and Cross-Layer Solutions for Mesh Networks

The vision of ubiquitous computing, in which digital assistance systems in the background provide support to the human user in a variety of life situations, is close to be realized due to the rapid development of microprocessor and software technologies [36]. Application scenarios can be found in building automation, industrial environments, or even in the medical field. Wirelessly connected mobile device assistance systems also provide intelligent services in public facilities and places that automatically provide contextual information to changing mobile subscribers.

Particularly in a dynamic environment, there are special requirements for the wireless networking of highly dynamic device ensembles. The changing subscribers constantly give rise to new situations and unforeseeable conditions in the network, e.g., with regard to the accessibility and connection quality of the communication nodes as well as their ad hoc networking and ad hoc cooperation. The devices and their interaction have to adapt spontaneously and independently to the changed conditions. In this regard, a meshed network topology is useful to allow an extensive and spontaneous data exchange over multi-hop and redundant paths, as is realized uniformly by mesh networks according to the standard IEEE 802.11s [37]. IEEE 802.11s is a sub-specification of the Wireless Local Area Network (WLAN) standard IEEE 802.11 since 2012 and enables the vendor-independent, cross-platform setup of a wireless, meshed network with the already existing variety of WLAN-enabled devices.

However, beyond the standard mesh functionality, there are still many requirements and issues for efficient meshed network communication that are not addressed by the 802.11s standard. Since 802.11s is an extension of the WLAN MAC layer specification, the mesh functionality is fully implemented at the link layer. It is therefore transparent for higher layers and applications. Hence, the properties of the underlying mesh network are not taken into account by layers and applications on top of a mesh network without further ado. Mechanisms such as flow and overload control, the choice of communication end points, or the parametrization of the data to be transmitted are usually not optimal, since the majority of existing protocols and applications are not designed for wireless mesh networks. In this chapter, a mesh network management is therefore described that is subsequently used to enable cross-layer collaboration between 802.11s and the P2P protocol BT (cf. **paper E** included in this thesis).

4.1 WLAN Mesh Networks Basing on IEEE 802.11s

The IEEE 802.11s standard was adopted in September 2011 and since 2012 has been part of the IEEE 802.11-2012 basic standard [37, 38, 39]. Unlike previous WLAN mesh solutions, the routing, i.e, the targeted forwarding of frames between the nodes takes place at layer 2 of the International Organization for Standardization (ISO)/Open Systems

Interconnection (OSI) model (see Figure 4.1). By fully integrating the mesh capabilities into the 802.11 MAC layer specification, it becomes transparent to higher layers. It can hence consider the characteristics of the WLAN technology directly. Thus, for example, information about the quality of the radio connection is available for MAC layer routing, referred to as path selection, and existing WLAN mechanisms for channel access and addressing as well as existing frame formats can be reused. Mesh nodes must use the same path selection method to form a common network. The 802.11s standard requires support for the Hybrid Wireless Mesh Protocol (HWMP) and associated ALM as part of a base profile. HWMP is a hybrid path selection protocol that allows a combination of reactive and proactive routing approaches. The default reactive mode is used to dynamically create paths between the mesh nodes. This mode has to be supported by each node. In the optional proactive mode, a paths are proactively established to a root node.

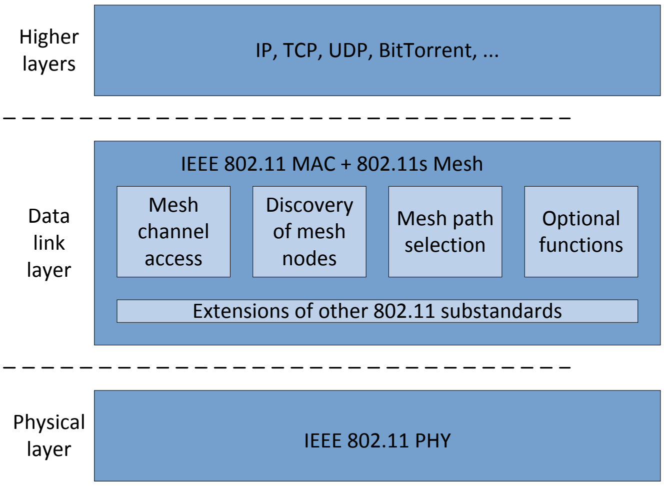


Figure 4.1: IEEE 802.11s at data link layer of the ISO/OSI model.

The ALM (see Equation 3) is a path selection metric designed specifically for use in the MAC layer.

$$c_a = \left[O + \frac{B_t}{r} \right] \cdot \frac{1}{1 - e_{fr}} \tag{3}$$

It specifies the time costs for the transmission of a test frame on a link (c_a) and, in addition to the overhead of the respective radio technology (O), also takes into account the bit error probability of the transmitted test frame (e_{fr}) for a given frame size (B_t) and data rate (r). Since ALM is a cumulative metric, the total cost of a path is the sum of the cost of all links of the path.

The scope of the 802.11s standard further specifies the basic mechanisms for channel access, discovery of mesh nodes, and spontaneous connection setup at the WLAN MAC layer. Other optional functions have also been defined in the standard, such as the proactive HWMP mode or collaborative channel switching.

4.2 Mesh Network Management

In a collaborative effort involving the author of this habilitation thesis, a mesh management solution based on the 802.11s reference implementation open80211s and various Linux userspace tools for manipulating network interfaces and settings was developed for 802.11s networks (see Figure 4.2) [40]. Consisting of an interchangeable platform-dependent (Linux userpace tools and kernel and open80211s) and an abstracting, platform-independent software part in Java, it implements functions for the autonomization of mesh nodes and their remote management. On the one hand, the automatic formation of a mesh network, referred to as bootstrapping, is performed by the software on each node referred to as mesh agent. On the other hand, a special node role called mesh manager performs additional functions such as Dynamic Host Configuration Protocol (DHCP) address assignment and allows the central status query as well as remote configuration of the agents via Simple Network Management Protocol (SNMP). For this purpose, a separate Management Information Base (MIB) data model for IEEE 802.11s has been defined, which maps the status information and configuration functions of the nodes. The management data of a node such as its network interfaces, mesh links and paths together with associated quality information or currently set 802.11s specific pa-

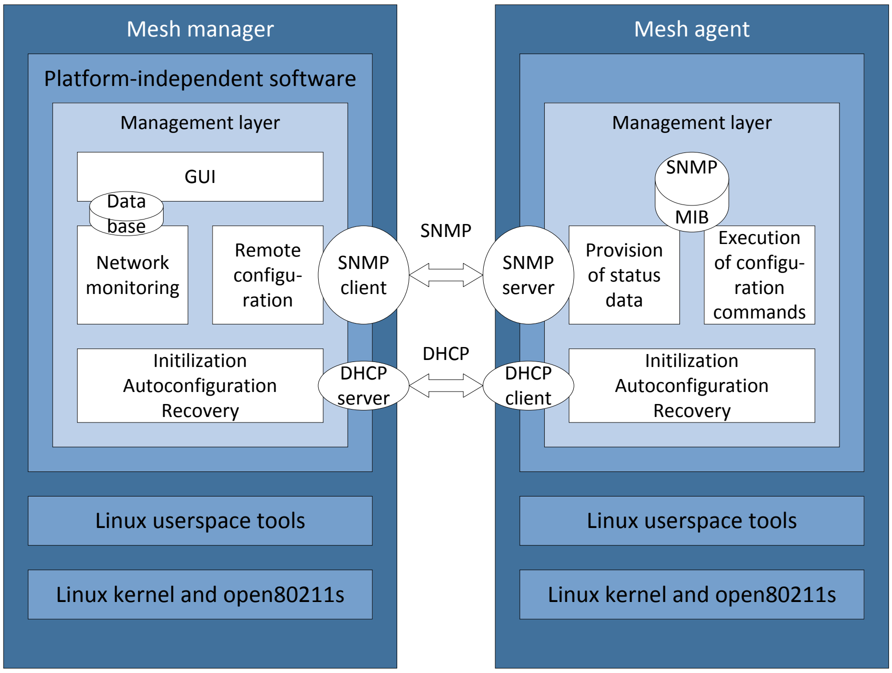


Figure 4.2: Composition of the developed mesh management solution.

rameters are included in the SNMP data model. The configuration functions include, for example, joining a mesh network, changing WLAN channels or modifying existing mesh paths, and changing the 802.11s specific parameters.

By determining the network-wide status information, a node in the role of the mesh manager receives a global view that is not available to the individual mesh nodes due to the characteristic of the path selection because of the default reactive mode of HWMP defined in the 802.11s standard. However, tests with an increasing number of devices showed that a distribution of the hitherto centralized mesh manager is indispensable for its scalability and reliability. Therefore, a first approach to the distribution of the centralized solution was investigated in [41].

In the subsequent section, cross-layer collaboration strategies are described, which leverage the knowledge provided by mesh agents, which locally run on each node as part of the developed mesh management solution.

4.3 Cross-Layer Collaboration between IEEE 802.11s and BT Networks

According to the ISO/OSI layer model, lower layers are transparent to the ones above. However, the more the physical underlay network differs from the application scenario, for which a higher-level process was designed, the greater the performance impact if the lower layers are ignored. In general, strong customizations are required for WLAN mesh networks, as previous protocols and higher layer applications are rather optimized for wired networks like Ethernet.

A suitable example for implementing distributed applications are P2P protocols, which form a logical distributed overlay network above the underlay. BT is a P2P protocol for collaborative data exchange [42]. Information to be exchanged is split into file parts of a fixed size called pieces and distributed to interested peers. BT is thus also very well suited for the efficient distribution of information of interest to a particular group of nodes in a mesh network, e.g., firmware updates. Another use case is the synchronization of selected status and control information between management nodes for distributed management and optimization of mesh networks. BT uses TCP as a transport layer and already benefits from optimization at this layer. BT also includes its own peer selection mechanism for selecting communication endpoints from the group of all participants interested in a collection of data referred to as torrent. The so-called choking algorithm restricts the upload to peers or allows them to download. Following the tit-for-tat principle, peers with the largest amount of data sent to other nodes receive uploads in return.

The choking algorithm in its standard version is designed for a wired network infrastructure. It takes neither physical proximity nor quality parameters of the wireless connections of a wireless mesh underlay into account. A transfer to nodes that are many hops away from each other leads to unnecessary load on the forwarding intermediate stations with regard to additional channel accesses and the forwarding time of the additional hops. End points of heavily loaded connections represented by a large ALM metric on the WLAN link layer are not perceived as such in the peer selection at the overlay and may be selected inconveniently.

Consequently, **paper E** included in this thesis proposes a cross-layer collaboration between IEEE 802.11s and BT networks. Using the mesh management solution as a middleware, the collaboration consists in using path quality and physical proximity information from the 802.11s mesh network in the peer selection algorithm of BT for improved network utilization and reduced overall distribution time of the data to all interested peers. Available upload slots are assigned to peers with the best ALM metrics from a mesh point of view. In order to provide BT with the ALM, a plugin for the open-source BT client Vuze has been written so that the choking algorithm can consider the ALM in the peer selection. Investigations in a real test environment show that taking into account the metric and the associated localization of the data exchange, the average distribution time can be reduced by 20% compared to default BT. To the best of the author's knowledge, the presented cross-layer solution is the first approach that integrates ALM into BT without changing the 802.11s routing protocol HWMP.

As part of future work, further cross-layer collaboration mechanisms for the BT network and networks at other layers will be examined.

4.4 Summary and Outlook

The IEEE 802.11s standard enables the wireless spontaneous networking of wireless devices in a uniform and flexible manner at link layer without intervening with the routing functionality at network layer. However, 802.11s lacks several functions for distributed control and optimization of the network. The presented management solution aims to continuously monitor the status of the device network without generating significant monitoring overhead and is also able to provide local information on a node. Building on this, online optimizations of the whole network or of applications running on single node can be carried out in the background on the basis of the status information obtained. These optimizations should, according to a cross-layer principle, take into account the behavior of applications and configure the physical network accordingly, as well as provide information from the lower network layers to higher layers in order to adapt the communication behavior to the given network structure and quality.

As demonstrated in this chapter, the challenges are already high in static networks when it comes to ensuring reliable and high-performance wireless communication, and it is important to take appropriate measures such as cross-layer collaboration. The challenges continue to grow as network participants continuously move. The effects of mobility are therefore examined in the following chapter. Moreover, as mentioned in Chapter 5, communication in the prospective IIoT requires RT capability. The vision of the IIoT also focuses on wireless RT communication, which can be realized with single-hop WLAN and multi-hop mesh networks. Compared to single-hop WLAN networks, mesh networks dispense with base stations to increase resilience and scalability and are thus suitable for networks at a larger scale. However, the RT capability must be ensured without a coordinating base station in mesh networks. There are only a few suggestions in the literature for building RT WLAN mesh networks. For instance, the work in [43] introduces the multi-hop TDMA mechanism called LiT MAC, which is based on standard WLAN hardware with modified driver software. LiT MAC addresses the challenges of implementing RT communication in multi-hop networks. Among other things, this requires efficient mechanisms for time synchronization, the distribution of the TDMA schedule to all subscribers, multi-channel support, simple integration of routing methods and the use of available space (SDMA),i.e., separated collision domains due to spatial distance of communication partners to ensure high performance. Since every realistic wireless environment has packet losses due to fading and interference, LiT MAC uses periodic retransmissions of control information, so eventually all nodes follow the same schedule. LiT MAC integrates various routing metrics and examines their stability over the established routes at runtime. The stability of the routes is, in addition to the performance of the data link layer, crucial for the quality of service of the application. The results show that a synchronization accuracy of the order of microseconds over multiple hops can be achieved and each subscriber can be assigned time slots of the order of 2 ms. Moreover, this is achieved by solely introducing low control overhead. Experiments in a nine-node test environment demonstrate that the LiT MAC approach enables robust operation despite external interference. Open research questions are how to reduce the time slots to less than 2 ms. Furthermore, it is necessary to investigate how the reconfiguration time of several seconds can be reduced in order to react to node churn at runtime.

5 Mobile Distributed Applications in Opportunistic Networks

After exploring the effects of the wireless nature of communication links in wireless mesh networks in the previous chapter, this chapter extends the topic by the mobility of network participants enabled by wireless communication. The proliferation of mobile phones means that communication in mobile networks is playing an increasingly important role, since in principle mobile users are able to consume information at any time in areas where mobile network operators ensure network coverage. As a consequence from this development, it is predicted for 2019 that mobile traffic will already account for 24 EB [44]. The original cellular network infrastructure, which was designed to carry voice data, will reach its limits as this amount of data is transported, and the current design of the network infrastructure must be reconsidered. In the long term, the solution proposed is Device-to-Device (D2D) communication to complement traditional mobile networks. Instead of installing expensive new infrastructure, D2D communication takes advantage of existing users' mobile devices or other easy-to-integrate equipment. If mobile nodes communicate by means of D2D communication, they opportunistically exchange data whenever they come in direct communication range instead of using the cellular infrastructure [45].

In this chapter, the fundamental limits and possibilities of a dynamic network of mobile nodes represented by pedestrians equipped with mobile phones are examined under various conditions in order to evaluate its applicability and performance. As a prerequisite, it is assumed that there are tasks that the mobile devices can complete cooperatively such as collecting and aggregating data. After having processed this data, the nodes can distribute the results to other interested nodes. This procedure describes a collaborative, participatory, and cooperative sensing and is often referred to as mobile cloud computing. The challenge is to reliably carry out the distributed computation under varying operating conditions with moving nodes and high churn.

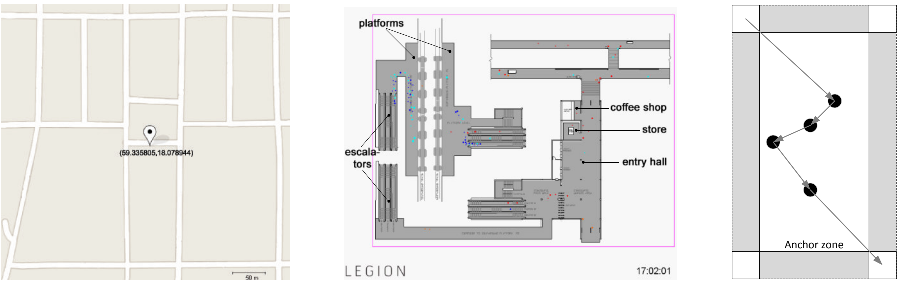
Two scenarios for dynamic environments are used as lead scenarios:

1. **Paper F** included in this thesis addresses distributed voting: This scenario aims at the distributed computation of the frequencies of the different answers when polling a crowd, e.g., for exit polls at elections.
2. **Paper G** included in this thesis addresses distributed crowd counting: The function to estimate the size of a dynamically formed crowd is important for the system. It can also be used to monitor crowding in environments for comfort and for avoiding emergencies.

5.1 Modeling Mobility

In order to realistically recreate pedestrian mobility, the Walkers traces [46] captured in Legion Studio [47] are used, a commercial simulator initially developed for designing and dimensioning large-scale spaces via simulation of pedestrian behaviors. Its multi-agent pedestrian model is based on advanced analytical and empirical models which have been calibrated by measurement studies. Please note that it is not possible to capture all states of human mobility with a single setup, however the scenarios are representative of typical daytime pedestrian mobility.

Each simulation run results in a trace file, containing a snapshot of the positions of all nodes in the system every 0.6 s. Fig. 5.1(a) and 5.1(b) present the scenarios: an outdoor urban scenario, modeling the Östermalm area of central Stockholm, and an indoor scenario, recreating a two-level subway station. The Östermalm scenario consists of a grid of interconnected streets. Fourteen passages connect the observed area to the outside world. The active area, i.e., the total surface of the streets, is 5872 m^2. The nodes are constantly moving, hence the scenario can be characterized as a high mobility scenario.



(a) (b) (c)

Figure 5.1: Urban scenarios: (a) a grid of streets representing the Östermalm scenario, (b) a two-level subway station, and (c) city square mobility model with random waypoint node movement in the anchor zone.

The Subway station has train platforms connected via escalators to the entry-level. Nodes arrive from any of five entries, or when a train arrives at the platform. The train arrivals create burstiness in the node arrivals and departures. Nodes congregate while waiting for a train at one of the platforms, or while taking a break in the store or the coffee shop at the entry level. The active area is 1921 m^2. If not stated otherwise, the input parameters of the Östermalm and the Subway scenario result in approximately the same mean node density of 0.1 nodes/m^2. For more information, the interested reader is referred to [48].

In addition to the realistic pedestrian mobility model, the synthetic city square model is used in the crowd counting scenario [49]. This open random waypoint model describes the movement of nodes in a confined area such as a city square. The main difference between this form of random waypoint model and the original model, in which no nodes leave the area, is the varying population over time. A node arrives to any of the four entry points according to a Poisson process, and immediately moves to a random point in the anchor zone. At a waypoint, a node may choose to exit the area with a certain probability, or to move to another randomly chosen waypoint inside the area. In the setup, nodes move in a rectangular area of size 5000 m^2, Figure 5.1(c). Nodes move at a speed of 1.3 m/s without pausing between two consecutive waypoints.

5.2 Distributed Voting

In the context of distributed voting in urban environments, a protocol called DiVote has been developed (cf. **paper F** included in this thesis). In general, the information obtained during a poll can be processed either in a centralized or in a decentralized manner. Centralized processing requires nodes to submit their votes to a central entity. However, this approach lacks scalability and poses privacy concerns as users might in general not want their votes to be seen by a central entity [50]. In particular, this may be of high importance in countries where people do not trust authorities. Contrary, decentralized (or distributed) processing requires nodes to compute local estimates of the result based on partial system knowledge.

DiVote operates at the application layer and has been developed to comply with the following requirements: be scalable, have fast convergence and high accuracy, and preserve node privacy. Each DiVote node can cast a binary vote (0 or 1) to a poll. For nodes to be able to calculate the anticipated global vote, they need to keep track of the votes of other peers in their vicinity throughout their lifetime. However, keeping track of a simple moving average may result in votes being counted multiple times if a node and a peer come in communication range more than once. Furthermore, keeping track of votes in the form of a binary vector may be consuming a lot of resources, especially if the vector contains long sequences of 0s or 1s. To address these problems, DiVote leverages the concept of D-GAP compression [51]. D-GAP compression provides a compressed representation of bit vectors in the form of integer vectors (later referred to as D-GAP vectors) and can be treated as a specialized variant of run length encoding. Each integer in a DGAP vector represents the number of consecutive 0s or 1s that are present in the bit vector at a given position. Whether the integer corresponds to a sequence of 0s or 1s is determined by the leading bit of the D-GAP vector. A leading bit of 0 shows that the first integer corresponds to a number of consecutive 0s, followed by a number of consecutive 1s and so on; a leading bit of 1 indicates the opposite behavior. With DiVote, each node locally keeps track of nodes it has obtained knowledge of, either directly or via other peers, as well as of their votes. This information is presented in the form of two correlated D-GAP vectors. Whenever a node first casts a vote, it adds itself to the shared-nodes vector, and it adds its vote to the votes vector. Observe that due to privacy preservation reasons, the position, in which information is stored in each of the D-GAP vectors, is determined by a cryptographic hash function such as MD5, which is calculated over a unique identifier, e.g., the node's MAC address.

It is assumed that all nodes carry devices and all are participating in the distributed poll in the area. In the scenario of distributed voting/polling, each node casts a binary vote v

= {0; 1} upon entry in the simulation, and votes are distributed according to a distribution f(x) with a mean E(x). For the evaluation, an implementation of an opportunistic content distribution system in the OMNeT++ simulator is used [52]. Each simulation run is executed in synchronous rounds of 0.6 s which corresponds to the granularity of the mobility traces used. Nodes broadcast their information at the beginning of each round[1]. To avoid collisions on the wireless medium, the broadcast transmission of each node in each round is distributed uniformly at random U(0; 0.5) s. The transmission range is set to 10 m.

The focus is on evaluating the following performance metrics.

Deviation: The deviation is a measure of the accuracy of the DiVote protocol, i.e., it shows how close the local estimate of a node is to the anticipated global result.

Compression ratio: The compression ratio is a measure of the efficiency and scalability of the DiVote protocol in terms of resource management, i.e., how much storage space does the protocol require for performing distributed voting computations in a mobile environment.

Information overhead: The information overhead is a measure of the processing load reduction for a node in the system and therefore indicates scalability as well. It shows how many of the received broadcasts do not need to be processed.

Simulation results for different arrival rates λ, two different scenarios, and two different distributions of nodes voting for one have been investigated. The results show that DiVote is suitable for operation in dynamic environments with high node mobility and that it scales well with the number of nodes in the system. DiVote demonstrates both fast convergence and high accuracy, with local estimates deviating at most by 3 % from the global value in dense scenarios. DiVote is able to achieve at least 19 % compression for realistic vote distributions, which makes it very probable that necessary voting information can be transmitted in only one broadcast message at a time. DiVote exhibits low processing load at the application layer as it requires only a fraction of the received broadcast messages (34 % in dense scenarios and even less in sparser scenarios) to be processed to achieve accurate local estimates.

Above, the main characteristics of a protocol suitable for distributed aggregation in dynamic environments were outlined. When comparing DiVote to state-of-the-art tree-based, restarted, and bookkeeping gossip-based protocols with respect to these characteristics, it reveals to be superior to these approaches.

Prospectively, it is planned to theoretically analyze DiVote and compare it with other existing schemes. The impact of technology and environment effects as well as of real network stacks shall be analyzed more precisely as part of future work.

---

[1] As proof-of-concept, apps for message broadcast between smartphones with Android and iOS using Bluetooth LE and Wi-Fi (Wi-Fi P2P on Android) have been implemented.

5.3 Distributed Crowd Counting

The most common approach for crowd counting today is the utilization of observation cameras. However, this approach has a number of drawbacks: it requires line-of-sight for operation, it is subject to quality degradation in the case of overlapping objects or changing environmental conditions, and it poses privacy issues. Other approaches rely on central entities, use measurements obtained from access points or dedicated infrastructure, or utilize different information acquired by human-carried devices for crowd counting. The author of this thesis has developed a protocol for crowd monitoring called UrbanCount, which operates at the application layer (cf. **paper G** included in this thesis). As opposed to the state-of-the art, UrbanCount is a fully distributed protocol for mobile crowd counting in urban areas and does not require infrastructure support. It is suitable for operation in urban environments with high node mobility and churn, i.e, the rate, with which nodes enter and leave the network. Furthermore, it is able to produce precise estimates under realistic mobility and to scale to support thousands of nodes.

As done for the scenario of DiVote, D-GAP compression was utilized to keep track of the overall crowd size in an area. However, with UrbanCount, each node not only locally keeps track of nodes it has obtained knowledge of. As crowds are characterized by churn, i.e., nodes entering and leaving the area, each node needs to inform others in its vicinity whenever it is about to exit an area. Note that on the one hand if a node begins to announce its intention to leave too early it might be removed too early from the estimate of other nodes. On the other hand, if it announce its leaving intention too late, this may lead to wrongful crowd estimations since no other node might overhear it. Hence, the time interval a node begins to announce its intention to leave has to be set carefully. It is further referred to as a farewell time. In UrbanCount, information is represented by two D-GAP vectors: a shared-nodes vector and a left-nodes vector. Upon entry into the area, a node inserts itself into its shared-nodes vector and its left-nodes vector remains empty. Note that the node inserts itself to the left-nodes vector whenever it reaches the beginning of its farewell time interval.

The simulation setup in OMNeT++ equals the the one of DiVote. The main performance metrics investigated is the accuracy, which is determined by means of the deviation. It shows how close the local estimate of a node is to the anticipated global result.

The performance of UrbanCount on the city square model and for realistic urban scenarios with respect to different arrival rates and various farewell times has been determined. The farewell time is varied as a multiple of the Inter-Contact Time (ICT). ICT is measured as the time that elapses from the beginning of one contact to the beginning of the next one [53]. In the first experiments, the ICT denotes the average ICT for all nodes in the system. In the final experiments, however, this assumption is released and a fully distributed solution is presented, in which nodes individually evaluate their perceived ICTs. Hence, the distributed system can adjust its operating conditions at runtime. Summarized, UrbanCount represents a fully distributed privacy-preserving protocol for crowd counting via device-to-device communication. It is shown that for densities above 0.1 nodes/m^2 distributed crowd counting produces precise estimates under realistic mobility. For these densities, it is shown that UrbanCount is scalable and achieves high accuracy of at least 98 % for synthetic and 93 % for realistic mobility scenarios.

As part of our future work, is is to be investigated how nodes can estimate in a distributed manner the density of an area in order to conclude on the reliability of their own local estimates.

5.4 Summary and Outlook

Distributed applications in dynamic opportunistic networks with highly dynamic participants have been shown to be able to be accurate and scalable. However, the dynamics due to mobility imposes tight constraints on the convergence speed of the algorithm. Since the mobile network nodes are only temporarily in communication range for a short time, information must be exchanged quickly. This happens by means of broadcast in the presented protocols, so that no connection establishment between nodes is necessary since for this the contact time is too short. The information disseminates epidemically, so that no guarantee can be given about the reliable distribution of information and that information is only sufficiently disseminated in networks with sufficiently high node density. It is questionable whether a different approach is feasible in the investigated highly dynamic scenarios, in which a connection can be established between nodes or adaptive routing through the mobile nodes is possible in order to transmit and distribute information in a more reliable and targeted manner.

Although the simulation results, obtained during the 1-year DFG research fellowship "A Reliable Distributed Computing System for Mobile Cloud Computing" of the author, are promising, as part of future work, it is planned to evaluate the simulated scenarios for distributed voting and crowd monitoring in a realistic testbed, including an investigation of interference and path-loss effects as well as communication technology-specific impacts. To facilitate the practical tests, the recently developed virtual prototyping framework called ViPMesh will be used instead of setting up a complex practical test bed [54]. It relies on WLAN interface emulation and QEMU-based system virtualization with nested container isolation to support early design analysis of real applications on top of an unmodified network stack. This approach allows the evaluation of common ad-hoc, infrastructure-mode, or mesh WLAN setups or any other MAC-layer variant, as configurable under Linux. Adopting an alternative time source approach for QEMU, ViPMesh acts as discrete-event simulator. Furthermore, it integrates comprehensive medium access, channel, and environment models with support for interference effects, IEEE 802.11n MIMO, multi-channel operation, and device mobility.

6 Summary of Original Work

This chapter contains a summary of the original work, which is presented in the habilitation thesis. The author of this thesis is first author of four of seven articles included here. Three Master's theses (J. Skodzik, M. Rethfeldt, E. B. Schweißguth), supervised by the writer of this thesis, have laid the foundation for paper B, C, and E, in which the author is second author. All papers that are included in the thesis appear in the same way they were published. However, minor formatting changes may apply.

At the end of the chapter, a list of further articles not included in this thesis is given. Only works are listed, which have been published or submitted after the completion of the author's PhD thesis in October 2012. Eight PhD theses completed (V. Altmann, J. Rohrbeck, J. Skodzik) or to be completed (B. Beichler, B. Konieczek, H. Puttnies, M. Rethfeldt, E. B. Schweißguth) have their origins in the publications, in which the author participated.

Paper A: Survey on Real-Time Communication Via Ethernet in Industrial Automation Environments

Peter Danielis, Jan Skodzik, Vlado Altmann, Eike B. Schweissguth, Frank Golatowski, Dirk Timmermann, and Joerg Schacht

In Proceedings of the 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8, `https://doi.org/10.1109/ETFA.2014.7005074`*, ISBN: 978-1-4799-4846-8, September 2014*

Abstract: For companies in the automation industry, the development of real-time Ethernet to connect devices is of high economic interest to replace conventional fieldbus systems. Therefore, many approaches for adapting Ethernet to real-time requirements come from industrial applications. This is a challenging task as the original Ethernet standard IEEE 802.3 was not designed for real-time data transmission. Likewise, protocols basing on Ethernet like TCP, UDP, and IP do typically not consider real-time requirements. Hence, adaptations on several OSI layers become necessary to make the industrial system meet hard real-time requirements. For this purpose, a multitude of real-time capable Industrial Ethernet systems has been developed, which solve the problems of standard Ethernet- and TCP/IP- or UDP/IP-based communication in a variety of ways. This paper gives a summary of different Industrial Ethernet protocols for the real-time data transmission via Ethernet in automation environments. Advantages and disadvantages of these protocols are analyzed with regard to their sustainability in terms of their real-time capability, reliability, scalability, self-configuration of the network, and hardware requirements. Against the background of connected devices tremendously growing in number and computational power in the prospective "Industrial Internet", consequences for future developments are drawn.

Contribution: The author of this thesis collected the material presented in this paper.

The writing of the paper was done in collaboration between the first three authors. All authors helped to check the collected information.

Paper B: HaRTKad: A Hard Real-Time Kademlia Approach

Jan Skodzik, Peter Danielis, Vlado Altmann, and Dirk Timmermann

In Proceedings of the 14th Annual IEEE Consumer Communications & Networking Conference (CCNC), pp. 309-314, `https://doi.org/10.1109/CCNC.2014.6866588`*, ISBN: 978-1-4799-2356-4, January 2014*

Abstract: The Internet of Things is becoming more and more relevant in industrial environments. As the industry itself has different requirements like (hard) real-time behavior for many scenarios, different solutions are needed to fulfill future challenges. Common Industrial Ethernet solution often leak scalability, flexibility, and robustness. Most realizations also require special hardware to guarantee a hard real-time behavior. Therefore, an approach is presented to realize a hard real-time network for automation scenarios using Peer-to-Peer technology. Kad as an implementation variant of the structured decentralized P2P protocol Kademlia has been chosen as base for the realization. As Kad is not suitable for hard real-time applications per se, changes of the protocol are necessary. Kad is extended by a TDMA-based mechanism. Additionally, to evaluate the performance an prototype is presented, which is realized on an embedded platform with a real-time operating system. Thereby, with the presented approach and a realized prototype it is possible to investigate the performance of a Kad network with hard real-time capabilities.

Contribution: The idea of this paper is an advancement of the first author's Master's thesis, supervised by the writer of this thesis, and is based on the fruitful discussions between the authors. The writing of the paper was done in collaboration between the first three authors. All authors helped to review content and the quality of presentation.

Paper C: ILP-Based Joint Routing and Scheduling for Time-Triggered Networks

Eike B. Schweissguth, Peter Danielis, Dirk Timmermann, Helge Parzyjegla, and Gero Mühl

In Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS), pp. 8-17, `https://doi.org/10.1145/3139258.3139289`*, ISBN: 978-1-4503-5286-4, October 2017*

Abstract: Networks in the automotive and aerospace area as well as in production facilities have to support time-critical (i.e., hard real-time) communication. For such applications, time-triggered Ethernet-based networking solutions provide the required timeliness, i.e., reliable packet delivery with deterministic latencies and low jitter. However, the routing and scheduling of the time-triggered traffic is an NP-hard problem. Hence, existing solutions to this problem make certain abstractions to reduce complexity if necessary. Nonetheless, such abstractions exclude feasible routing and scheduling options from the design space. Specifically, it is a typical approach to assume a fixed routing and model the scheduling problem, only. Therefore, we present a novel ILP formulation that can be used to jointly solve the routing and scheduling problem for time-triggered Ethernet networks. Using this formulation, it is possible to solve various scheduling problems that are infeasible when using a fixed shortest path routing with separate scheduling. Compared to a fixed load balanced routing with separate scheduling, schedules computed with our formulation can offer lower communication latencies.

Contribution: The idea of this paper is an advancement of the first author's Master's thesis, supervised by the writer of this thesis, and is based on the fruitful discussions between the authors. The writing of the paper was done in collaboration between the first, second, and fourth author. All authors helped to review content and the quality of presentation.

Paper D: Dynamic Flow Migration for Delay Constrained Traffic in Software-Defined Networks

Peter Danielis, György Dán, James Gross and André Berger

In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM) 2017, pp. 1-7, `https://doi.org/10.1109/GLOCOM.2017.8254753`, ISBN: 978-1-5090-5019-2, December 2017

Abstract: Various industrial control applications have stringent end-to-end latency requirements in the order of a few milliseconds. Software-defined networking (SDN) is a promising solution in order to meet these stringent requirements under varying traffic patterns, as it enables the flexible management of flows across the network. Thus, SDN allows to ensure that traffic flows use congestion-free paths, reducing the delay to forwarding and processing delays at the SDN nodes. However, accommodating new flows at runtime is under such a setting challenging as it may require the migration of existing flows, without interrupting ongoing traffic. In this paper, we consider the problem of dynamic flow migration and propose a polynomial time algorithm that can find a solution if direct flow migration is feasible. We furthermore propose an algorithm for computing both direct and indirect flow migration and prove its correctness. Numerical results obtained on a FatTree network topology show that flow migration is typically necessary for networks with a moderate number of flows, while direct flow migration is feasible in around 60% of the cases.

Contribution: The idea of this paper is based on the fruitful discussions between the authors. The author of this thesis performed the evaluation presented in this paper. The writing of the paper was done in collaboration between all authors.

Paper E: Integration of QoS Parameters from IEEE 802.11s WLAN Mesh Networks into Logical P2P Overlays

Michael Rethfeldt, Peter Danielis, Björn Konieczek, Felix Uster, Dirk Timmermann

In Proceedings of the 14th IEEE International Conference on Ubiquitous Computing and Communications (IUCC), pp. 1170-1177, https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.175, ISBN: 978-1-5090-0153-8, Liverpool, England, Oktober 2015

Abstract: Adopted in late 2011, IEEE 802.11s comes as the first industry standard to enable vendor-independent and inter-operable WLAN mesh networks. Featuring automatic device interconnection and routing, they provide a higher scalability, flexibility, and robustness compared to common centralized WLAN infrastructures. The 802.11s standard defines mandatory support of the Hybrid Wireless Mesh Protocol (HWMP) and Airtime Link Metric (ALM) for MAC-layer routing. While 802.11s covers the physical network underlay, Peer-to-Peer (P2P) protocols are an equivalent on the application level. In contrast to centralized client/server communication, they establish a fail-safe and scalable logical network overlay for, e.g., distributed content sharing, streaming, search, or synchronization. Thus, P2P networks exhibit many of the characteristics of physical WLAN mesh networks. It is obvious to consider joint solutions, where both technologies are combined to leverage future distributed local area wireless applications. Nevertheless, common P2P protocols, such as BitTorrent, do not consider the structure of the physical underlay while performing topology management. Furthermore, they are primarily designed to be used over wired communication networks such as large parts of the Internet. When deployed over WLAN mesh networks with their quickly varying channel conditions, BitTorrent shows severe performance drawbacks. We present a cross-layer approach based on 802.11s and BitTorrent, that optimizes application layer peer selection by considering the mesh standard's routing metric ALM. Our solution was implemented and evaluated in a real-world test bed. Results show that average download time can be reduced by up to 20 % already in small network setups.

Contribution: The approach of this paper leverages the idea of the first author's Master's thesis, supervised by the writer of this thesis, and is based on the fruitful discussions between the authors. The writing of the paper was done in collaboration between the first two authors. All authors helped to review content and the quality of presentation.

Paper F: DiVote: A Distributed Voting Protocol for Mobile Device-to-Device Communication

Peter Danielis, Sylvia T. Kouyoumdjieva, and Gunnar Karlsson

In Proceedings of the International Teletraffic Congress 28 (ITC), pp. 69-77, `https://doi.org/10.1109/ITC-28.2016.118`*, ISBN:978-0-9883045-1-2, September 2016*

Abstract: Distributed aggregation algorithms have traditionally been applied to environments with no or rather low rates of node churn. The proliferation of mobile devices in recent years introduces high mobility and node churn to these environments, thus imposing a new dimension on the problem of distributed aggregation in terms of scalability and convergence speed. To address this, we present DiVote, a distributed voting protocol for mobile device-to-device communication. We investigate a particular use case, in which pedestrians equipped with mobile phones roam around in an urban area and participate in a distributed yes/no poll, which has both spatial and temporal relevance to the community. Each node casts a vote and collects votes from other participants in the system whenever in communication range; votes are immediately integrated into a local estimate. The objective of DiVote is to produce a precise mapping of the local estimate to the anticipated global voting result while preserving node privacy. Since mobile devices may have limited resources allocated for mobile sensing activities, DiVote utilizes D-GAP compression. We evaluate the proposed protocol via extensive trace-driven simulations of realistic pedestrian behavior, and demonstrate that it scales well with the number of nodes in the system. Furthermore, in densely populated areas the local estimate of participants does not deviate by more than 3 % from the global result. Finally, in certain scenarios the achievable compression rate of DiVote is at least 19 % for realistic vote distributions.

Contribution: The idea of this paper is based on the fruitful discussions between the authors. The author of this thesis performed the work presented in this paper. The writing of the paper was done in collaboration between all authors.

Paper G: UrbanCount: Mobile Crowd Counting in Urban Environments

Peter Danielis, Sylvia T. Kouyoumdjieva, and Gunnar Karlsson

In Proceedings of the 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 640-648, `https://doi.org/10.1109/IEMCON.2017.8117189`, ISBN: 978-1-5386-3371-7, October 2017 (Best Paper Award)

Abstract: Surveillance, management and estimation of spontaneous crowd formations in urban environments, e.g., during open-air festivals or rush hours, are necessary measures for city administration. Most solutions that implement these measures however require additional costly hardware installations (e.g., installation of observation cameras) and infrastructure support, and often pose privacy concerns. In this work, we present *UrbanCount*, a fully distributed crowd counting protocol for cities with high crowd densities. UrbanCount relies on mobile device-to-device communication to perform crowd estimation. Each node collects crowd size estimates from other participants in the system whenever in communication range and immediately integrates these estimates into a local estimate. The objective of UrbanCount is to produce a precise mapping of the local estimate to the anticipated global result while preserving node privacy. We evaluate the proposed protocol via extensive trace-driven simulations of synthetic *and* realistic mobility models. Furthermore, we investigate the dependency between accuracy and density, and demonstrate that in dense environments the local estimate does not deviate by more than 2% for synthetic and 7% for realistic scenarios.

Contribution: The idea of this paper is based on the fruitful discussions between the authors. The author of this thesis performed the work presented in this paper. The writing of the paper was done in collaboration between all authors.

Articles not included in this thesis:

- Henning Puttnies, Peter Danielis, Dirk Timmermann, *PTP-LP: Using Linear Programming to Increase the Synchronization Precision of IEEE 1588 PTP,* accepted for publication at the IEEE Global Telecommunications Conference (GLOBECOM), December 2018
- Christian Wernecke, Helge Parzyjegla, Gero Múhl, Peter Danielis, Dirk Timmermann, *Realizing Content-Based Publish/Subscribe with P4,* accepted for publication at the 4th IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN), November 2018
- Michael Rethfeldt, Benjamin Beichler, Peter Danielis, Tim Brockmann, Christian Haubelt, Dirk Timmermann, *CHaChA: Clustering Heuristic and Channel Assignment for IEEE 802.11s Mesh Networks,* accepted for publication at the 9th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), November 2018
- Michael Rethfeldt, Benjamin Beichler, Peter Danielis, Felix Uster, Christian Haubelt, Dirk Timmermann, *MeNTor: A Wireless-Mesh-Network-Aware Data Dissemination Overlay based on BitTorrent Ad Hoc Networks,* in Elsevier Ad Hoc Networks, Volume 79, pp. 146-159, `https://doi.org/10.1016/j.adhoc.2018.06.013`, ISSN: 1570-8705, Elsevier B. V., Amsterdam, The Netherlands, October 2018
- Michael Rethfeldt, Benjamin Beichler, Peter Danielis, Christian Haubelt, Dirk Timmermann, *Enabling the Management of IEEE 802.11s Wireless Mesh Networks,* accepted for publication at the 17th GI/ITG KuVS Fachgespräch Sensornetze (FGSN), September 2018
- Henning Puttnies, Peter Danielis, Leonard Thiele, Dirk Timmermann, *jUDPWrapper: A Lightweight Approach to Access the UDP Functionality of OMNeT++/INET from Java,* accepted for publication at the 5th OMNeT++ Community Summit, September 2018
- Henning Puttnies, Peter Danielis, Enkhtuvshin Janchivnyambuu, Dirk Timmermann, *A Simulation Model of IEEE 802.1AS gPTP for Clock Synchronization in OMNeT++,* accepted for publication at the 5th OMNeT++ Community Summit, September 2018
- Peter Danielis, Henning Puttnies, Eike Schweissguth, Dirk Timmermann, *Real-Time Capable Internet Technologies for Wired Communication in the Industrial IoT—a Survey,* accepted for publication at the 23rd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), September 2018
- Sylvia T. Kouyoumdjieva, Peter Danielis, Gunnar Karlsson, *Non-Image Based Approaches for People Counting: Taxonomy, Requirements and Open Challenges,* in Proc. 14th Swedish National Computer Networking Workshop (SNCNW), pp. 9-12, May 2018

- Fabian Hölzke, Peter Danielis, Frank Golatowski, Dirk Timmermann, *A Fusion Approach for the Localization of Humans in Factory Environments,* in Proc. 1st IEEE International Conference on Industrial Cyber-Physical Systems (ICPS), pp. 59-64, `https://doi.org/10.1109/ICPHYS.2018.8387638`, ISBN: 978-1-5386-6531-2, May 2018
- Sylvia T. Kouyoumdjieva, Peter Danielis, Gunnar Karlsson, *Survey of Non-Image Based Approaches for Counting People,* submitted to IEEE Communications Surveys and Tutorials, April 2018
- Michael Rethfeldt, Benjamin Beichler, Hannes Raddatz, Felix Uster, Peter Danielis, Christian Haubelt, Dirk Timmermann, *Mini-Mesh: Practical Assessment of a Miniaturized IEEE 802.11n/s Mesh Testbed,* in Proc. 2018 IEEE Wireless Communications and Networking Conference (WCNC), pp. 1-6, `https://doi.org/10.1109/WCNC.2018.8377247`, ISBN: 978-1-5386-1734-2, April 2018
- Peter Danielis, Sylvia T. Kouyoumdjieva, Gunnar Karlsson, *A Distributed Protocol for Crowd Counting in Urban Environments,* in Proc. 1st GI/ITG KuVS-Fachgespräch Fog Computing, pp. 13-16, http://repositum.tuwien.ac.at/obvutwoa/content/titleinfo/2514843, March 2018
- Arne Wall, Hannes Raddatz, Michael Rethfeldt, Peter Danielis, Dirk Timmermann, *Performance Evaluation of MAC-Layer Trust Zones over Virtual Network Interfaces,* in Proc. 4th Conference On Mobile And Secure Services (MobiSecServ), pp. 1-5, `https://doi.org/10.1109/MOBISECSERV.2018.8311442`, ISBN: 978-1-5386-3253-6, February 2018
- Arne Wall, Hannes Raddatz, Michael Rethfeldt, Peter Danielis, Dirk Timmermann, *ANTs: Application-Driven Network Trust Zones on MAC-Layer in Smart Buildings,* in Proc. 15th Annual IEEE Consumer Communications & Networking Conference (CCNC), pp. 1-2, `https://doi.org/10.1109/CCNC.2018.8319304`, ISBN: 978-1-5386-4790-5, January 2018
- Henning Puttnies, Peter Danielis, Christian Koch, Dirk Timmermann, *Java Extensions for OMNeT++ 5.0,* in Proc. 4th OMNeT++ Community Summit, pp. 38-42, `https://arxiv.org/abs/1709.02823`, September 2017
- Benjamin Beichler, Michael Rethfeldt, Hannes Raddatz, Björn Konieczek, Peter Danielis, Christian Haubelt, Dirk Timmermann, *Optimization of a novel WLAN Simulation Framework for Prototyping Network Applications and Protocols,* in Proc. GI / ITG / GMM Workshop - Methoden und Beschreibungssprachen zur Modellierung und Verifikation von Schaltungen und Systemen (MBMV), pp. 15-26, February 2017
- Henning Puttnies, Dirk Timmermann, Peter Danielis, *An Approach for Precise, Scalable, and Platform Independent Clock Synchronization,* in Proc. 14th Annual

IEEE Consumer Communications & Networking Conference (CCNC), pp. 461-466, `https://doi.org/10.1109/CCNC.2017.7983152`, ISBN: 978-1-5090-6196-9, January 2017

- Michael Rethfeldt, Hannes Raddatz, Benjamin Beichler, Björn Konieczek, Dirk Timmermann, Christian Haubelt, Peter Danielis, *ViPMesh: A Virtual Prototyping Framework for IEEE 802.11s Wireless Mesh Networks,* in Proc. 12th IEEE International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob), pp. 1-7, `https://doi.org/10.1109/WFCS.2016.7496529`, ISBN: 978-1-5090-0724-0, October 2016
- Henning Puttnies, Björn Konieczek, Jakob Heller, Dirk Timmermann, Peter Danielis, *Algorithmic Approach to Estimate Variant Software Latencies for Latency-Sensitive Networking,* in Proc. 7th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 1-7, `https://doi.org/10.1109/IEMCON.2016.7746281`, ISBN: 978-1-5090-0996-1, October 2016
- Björn Konieczek, Jan Skodzik, Peter Danielis, Vlado Altmann, Michael Rethfeldt, Dirk Timmermann, *HaRTKad: A P2P-based Concept for Deterministic Communication and Its Limitations,* in Proc. 21th IEEE Symposium on Computers and Communication (ISCC), pp. 1198-1203, `https://doi.org/10.1109/ISCC.2016.7543893`, ISBN: 978-1-5090-0679-3, June 2016
- Peter Danielis, Sylvia T. Kouyoumdjieva, Gunnar Karlsson, *A Protocol for Distributed Voting in Urban Environments,* in Proc. 12th Swedish National Computer Networking Workshop (SNCNW), pp. 50-53, June 2016
- Eike B. Schweissguth, Peter Danielis, Christoph Niemann, Dirk Timmermann, *Application-Aware Industrial Ethernet Based on an SDN-Supported TDMA Approach,* in Proc. 12th IEEE World Conference on Factory Communication Systems (WFCS), pp. 1-8, `https://doi.org/10.1109/WFCS.2016.7496496`, ISBN: 978-1-5090-2338-7, May 2016
- Michael Rethfeldt, Peter Danielis, Benjamin Beichler, Björn Konieczek, Felix Uster, Dirk Timmermann, *Evaluating Cross-Layer Cooperation of Congestion and Flow Control in IEEE 802.11s Networks,* in Proc. 30th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 181-188, `https://doi.org/10.1109/AINA.2016.12`, ISBN: 978-1-5090-1857-4, March 2016
- Michael Rethfeldt, Arne Wall, Peter Danielis, Björn Konieczek, Dirk Timmermann, *AKadeMesh: Software-Defined Overlay Adaptation for the Management of IEEE 802.11s Networks,* in Proc. IEEE Consumer Communications & Networking Conference (CCNC), pp. 477-482, `https://doi.org/10.1109/CCNC.2016.7444826`, ISBN: 978-1-4673-9292-1, January 2016
- Peter Danielis, Vlado Altmann, Jan Skodzik, Tim Wegner, Achim Koerner, Dirk Timmermann, *P-DONAS: A P2P-based Domain Name System in Access Networks,*

in ACM Transactions on Internet Technology, pp. 11:1-11:21, `https://doi.org/10.1145/2808229`, vol. 15, no. 3, ISSN: 1533-5399, September 2015

- Peter Danielis, Vlado Altmann, Jan Skodzik, Eike B. Schweissguth, Frank Golatowski, Dirk Timmermann, *Emulation of SDN-Supported Automation Networks* in Proc. 20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8, `https://doi.org/10.1109/ETFA.2015.7301517`, ISBN: 978-1-4673-7929-8/15, September 2015
- Peter Danielis, Jan Skodzik, Vlado Altmann, Frank Golatowski, Dirk Timmermann, *DuDE-Cloud: A Resilient High Performance Cloud,* in Proc. 20th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 28-34, `https://doi.org/10.1109/ETFA.2015.7301658`, ISBN: 978-1-4673-7929-8/15, September 2015
- Peter Danielis, Jan Skodzik, Vlado Altmann, Benjamin Kappel, Dirk Timmermann, *Extensive Analysis of the Kad-based Distributed Computing System DuDE,* in Proc. 20th IEEE Symposium on Computers and Communications (ISCC), pp. 1028-1033, `https://doi.org/10.1109/ISCC.2015.7405505`, ISBN: 978-1-4673-7194-0, July 2015
- Michael Rethfeldt, Peter Danielis, Guido Moritz, Björn Konieczek, Dirk Timmermann, *Design and Development of a Management Solution for Wireless Mesh Networks based on IEEE 802.11s,* in Proc. 14th IFIP/IEEE Symposium on Integrated Network and Service Management (IM), pp. 902-905, `https://doi.org/10.1109/INM.2015.7140405`, ISBN: 978-3-901882-76-0, May 2015
- Jan Skodzik, Peter Danielis, Vlado Altmann, Eike B. Schweissguth, Dirk Timmermann, *PSP-Auto: An DHT-based Storage and Retrieve System for Automation Scenarios,* in Proc. 9th Annual IEEE International Systems Conference (SysCon), pp. 853-858, `https://doi.org/10.1109/SYSCON.2015.7116857`, ISBN: 978-1-4799-5927-3, April 2015
- Jan Skodzik, Peter Danielis, Vlado Altmann, Björn Konieczek, Eike B. Schweissguth, Frank Golatowski, Dirk Timmermann, *CoHaRT: Deterministic Transmission of Large Data Amounts using CoAP and Kad,* in Proc. IEEE International Conference on Industrial Technology (ICIT), pp. 1851-1856, `https://doi.org/10.1109/ICIT.2015.7125366`, ISBN: 978-1-4799-7799-4, March 2015
- Peter Danielis, Jan Skodzik, Vlado Altmann, Lennard Lender, Dirk Timmermann, *Dynamic Search Tolerance at Runtime for Lookup Determinism in the DHT-based P2P Network Kad,* in Proc. IEEE Consumer Communications & Networking Conference (CCNC), pp. 357-362, `https://doi.org/10.1109/CCNC.2015.7158002`, ISBN: 978-1-4799-6389-8, January 2015
- Jan Skodzik, Vlado Altmann, Peter Danielis, Moritz Koal, Dirk Timmermann, *An Optimized WS-Eventing for Large-Scale Networks,* in Proc. 19th IEEE Interna-

tional Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-6, `https://doi.org/10.1109/ETFA.2014.7005248`, ISBN: 978-1-4799-4846-8, September 2014

- Vlado Altmann, Jan Skodzik, Peter Danielis, Johannes Müller, Frank Golatowski, Dirk Timmermann, *A DHT-based Scalable Approach for Device and Service Discovery,* in Proc. 12th IEEE International Conference on Embedded and Ubiquitous Computing (EUC), pp. 97-103, `https://doi.org/10.1109/EUC.2014.23`, ISBN: 978-0-7695-5249-1, August 2014
- Vlado Altmann, Jan Skodzik, Peter Danielis, Frank Golatowski, Dirk Timmermann, *Real-Time Capable Hardware-based Parser for Efficient XML Interchange,* in Proc. 9th IEEE/IET International Symposium on Communication Systems, Networks and Digital Signal Processing (CSNDSP), pp. 415-420, `https://doi.org/10.1109/CSNDSP.2014.6923861`, ISBN: 978-1-4799-2581-0, July 2014
- Jan Skodzik, Vlado Altmann, Peter Danielis, Arne Wall, Dirk Timmermann, *A Kad Prototype for Time Synchronization in Real-Time Automation Scenarios,* in Proc. World Telecommunications Congress (WTC), pp. 1-6, ISBN: 978-3-8007-3602-7, June 2014
- Jan Skodzik, Peter Danielis, Vlado Altmann, Dirk Timmermann, *Extensive Analysis of a Kad-based Distributed Storage System for Session Data,* in Proc. 18th IEEE Symposium on Computers and Communications (ISCC), pp. 489-494, `https://doi.org/10.1109/ISCC.2013.6754994`, ISBN: 978-1-4799-3755-4, July 2013
- Vlado Altmann, Peter Danielis, Jan Skodzik, Frank Golatowski, Dirk Timmermann, *Optimization of Ad Hoc Device and Service Discovery in Large Scale Networks,* in Proc. 18th IEEE Symposium on Computers and Communications (ISCC), pp. 833-838, `https://doi.org/10.1109/ISCC.2013.6755052`, ISBN: 978-1-4799-3755-4, July 2013
- Jan Skodzik, Peter Danielis, Vlado Altmann, Dirk Timmermann, *Time Synchronization in the DHT-based P2P Network Kad for Real-Time Automation Scenarios,* in Proc. 14th IEEE International Symposium and Workshops on a World of Wireless, Mobile and Multimedia Networks (WoWMoM), pp. 1-6, `https://doi.org/10.1109/WoWMoM.2013.6583490`, ISBN: 978-1-4673-5827-9, June 2013
- Jan Skodzik, Vlado Altmann, Benjamin Wagner, Peter Danielis, Dirk Timmermann, *A Highly Integrable FPGA-Based Runtime-Configurable Multilayer Perceptron,* in Proc. 27th IEEE International Conference on Advanced Information Networking and Applications (AINA), pp. 429-436, `https://doi.org/10.1109/AINA.2013.19`, ISSN:1550-445X, ISBN: 978-1-4673-5550-6, March 2013
- Vlado Altmann, Jens Rohrbeck, Jan Skodzik, Peter Danielis, Dirk Timmermann, Maik Rönnau, Matthias Ninnemann, *SWIFT: A Secure Web Domain Filter in Hardware,* in Proc. 7th International Symposium on Security and Multimodality

in Pervasive Environment (SMPE)), pp. 678-683, `https://doi.org/10.1109/WAINA.2013.15`, ISBN: 978-1-4673-6239-9, March 2013

7 Conclusions and Future Work

This habilitation thesis explored various possibilities for the network and application design of reliable distributed systems. Depending on the application, it was investigated how RT capability as well as high scalability, accuracy, robustness, and performance can be achieved.

First, approaches for RT capable Ethernet systems were investigated since existing solutions do not meet the requirements of future systems for the IIoT in all respects. A decentralized RT Ethernet system was developed, which is intended to remedy the limitations of existing systems and at the same time omits a central control entity. The next step was to explore the benefits of using a scheduling and routing algorithm executed by a central SDN controller to implement RT Ethernet networks. Finally, scalable algorithms were introduced that take into account dynamically changing RT requirements in Ethernet networks.

Subsequently, the focus was shifted to wireless IEEE 802.11s mesh networks with static network subscribers. For these, cross-layer collaboration mechanisms have been developed to better match the mesh link layer and overlay applications to increase application performance and network robustness. Finally, the last part is devoted to the consideration of dynamic opportunistic networks, in which mobile distributing applications are executed. It has been studied how to design these to ensure high scalability and accuracy for high mobility applications.

Conclusions

Survey on Real-Time Communication Via Ethernet in Industrial Automation Environments

It becomes apparent that none of the established IE systems meets all requirements concerning high reliability, scalability, flexibility in terms of self-configuration, and cost-effective standard Ethernet hardware. No currently established solution achieves isochronous RT data transmission without special hardware and no Single Point of Failure (SPoF) at the same time. The protocols at application layer introduced or applied by the established IE systems are optimized to the transmission of process data and do not allow for the reliable RT transmission of increasing amounts of data. Only TTEthernet offers degrees of freedom in terms of a protocol for the deterministic transmission of any data but does not specify a protocol either. The TSN technology currently under development may be able to meet all requirements in the future, but TSN-capable network components must be configured using algorithms that have to be specially developed.

HaRTKad: A ***H**ard **R**eal-**T**ime **Kad**emlia Approach*

Based on these findings, an approach was presented to realize a fully decentralized Kad network enabling hard real-time operations. Additionally, the prototype for a hard real-time Kad node called HaRTKad node was presented. The combination of the presented modified Kad protocol and a working prototype allows for the realization of hard RT applications with a high resilience, flexibility, and without any SPoF. Furthermore, administrative scalability and usability in adding and removing further instances is simplified as no managing central instance is necessary. Every node is able to act autonomously on its given data in a deterministic way to enable the hard real-time behavior.

ILP-Based Joint Routing and Scheduling for Time-Triggered Networks

A joint routing and scheduling algorithm was successfully integrated into an ILP formulation and its applicability for scheduling time-triggered Ethernet networks was verified. In its current state, the approach is usable for TTEthernet and TSN with time-aware traffic shaping, and it can be modified for scheduling of Profinet with minor effort. Test cases were created that enable a comparison of the joint routing and scheduling approach to other scheduling methods that use a fixed routing. Obtained results were analyzed with respect to two important aspects: schedulability and communication latency. In these test cases, generic traffic patterns were used with medium to high overall network load and realistic but highly demanding flow characteristics. For such scenarios, the fixed routing approaches either have drawbacks with respect to schedulability or latencies, while the joint routing and scheduling algorithm yields good results with respect to both of these aspects in all test cases. The scheduling approach with fixed load balanced routing could keep up with joint routing and scheduling in terms of schedulability, but showed up to 61.8 % higher flow latencies. Contrary, the scheduling approach with fixed shortest path routing could provide good latency results, but in the worst case, only 33.3 % of the traffic patterns that were schedulable by joint routing and scheduling could be successfully scheduled. Based on the results, it becomes clear that a comprehensive modeling approach like joint routing and scheduling is the method of choice for routing and scheduling of time-triggered networks.

Dynamic Flow Migration for Delay Constrained Traffic in Software-Defined Networks

The problem of dynamic flow migration in software-defined networks was addressed. Two algorithms were developed that allow conflict-free direct and indirect flow migration and thus enable inserting an additional flow at runtime. The proposed algorithm for direct flow migration runs in polynomial time but does not always find a solution to the flow migration problem. The generic algorithm finds all solutions to the flow migration problem in case it is feasible at all, but at the price of an increased computational complexity. The results show that flow migration is required in 24 % up to 30 % of all cases for a FatTree topology with moderate traffic, and the necessary flow migrations typically require several migration steps. For the investigated topologies, direct flow migration is possible in 62 % up to 92 % of the cases.

Integration of QoS Parameters from IEEE 802.11s WLAN Mesh Networks into Logical P2P Overlays

A developed mesh management framework was used as a middle-ware for exploring cross-layer optimization strategies within the specification boundaries of the 802.11s standard.

A bottom-up approach was pursued that performs integration of the 802.11s default MAC-layer ALM into the application layer to improve the BT P2P protocol and its choking algorithm, running on top of the physical mesh network. Defined as mandatory default metric in 802.11s, ALM is guaranteed to be available on every standard-compliant mesh node. The presented cross-layer solution is the first approach to integrate ALM into BT while no changes to the 802.11s routing protocol HWMP are required. As a logical starting point, plain ALM is passed to BT's choking algorithm to replace the upload rate as default peer selection criterion. Directly using ALM shows improvement already in a small static mesh setup with eight nodes and average file download time on each node is reduced by around 4 %. Consequently, ALM is combined with a neighborhood scope limit. In the test bed, the second approach reduces average download time by up to 20 %, by still maintaining interoperability to the 802.11s standard and HWMP. The presented mesh setup represents a decisive step towards the mitigation of the mismatch between logical overlays and wireless physical underlays and hence forms the basis for establishing an even more comprehensive test bed. The trend of achievable download time reduction is clearly visible using this setup and fixed parameter set (BT defaults).

DiVote: A Distributed Voting Protocol for Mobile Device-to-Device Communication

DiVote was presented, which is a distributed voting protocol in the context of urban polling, which is suitable for environments in which nodes exhibit high mobility. DiVote relies on device-to-device communication to exchange voting information. The proposed DiVote protocol exhibits the following main features:

- Privacy: By using a cryptographic hash function, votes cannot be related to the corresponding node identities.
- Convergence speed: The dynamism due to mobility imposes tight constraints on the convergence speed of the algorithm. Consequently, DiVote immediately updates the local estimate. Simulation results obtained when applying DiVote to realistic pedestrian mobility traces show that even in sparse scenarios local estimates quickly converge to the global result.
- Accuracy: At the same time, accuracy of local estimates is ensured as DiVote avoids to erroneously count votes multiple times, which is of decisive importance as the same node may be encountered several times. In dense scenarios, the local estimate does not deviate by more than 3 % from the global result after the system reaches the steady state.
- Scalability: Rather than storing shared nodes and their votes in plain bit vectors, DiVote uses D-GAP compression to be scalable in terms of required storage capacity. For realistic voting distributions, at least 19 % compression is achieved. Furthermore, the processing overhead introduced at the application layer is very low as only a fraction of the received messages (approximately 30 %) has to be processed even in the densely populated scenarios.

UrbanCount: Mobile Crowd Counting in Urban Environments

Surveillance, management, and estimation of spontaneous crowd formations in urban environments, e.g., during open-air festivals or rush hours, are necessary measures for city administration. UrbanCount was presented, which is a fully distributed protocol for crowd counting via device-to-device communication, which is suitable for operation in urban environments with high node mobility and churn. Each node collects crowd size estimates from other participants in the system whenever in direct communication range and immediately integrates these estimates into a local estimate. The proposed protocol was evaluated via extensive trace-driven simulations of synthetic and realistic mobility models, and two main questions were answered: when is distributed counting possible, and how accurate can it be. For densities above 0.1 nodes/m^2 distributed crowd counting was shown to be able to produce precise estimates regardless of the area in which mobility occurs. When the density is sufficient, UrbanCount is able to produce a local estimate with at least 98% accuracy for synthetic and 93% for realistic mobility scenarios. Furthermore, UrbanCount provides a scalable solution for crowd counting for both indoor and outdoor scenarios. Finally, the performance of UrbanCount is not affected if the characteristics of mobility in an area are not known in advance. As part of our future work, we plan to evaluate other scenarios in a realistic testbed, including an investigation of interference and path-loss effects as well as communication technology-specific impacts.

Future Research Directions

It can be concluded that there is a need for the advancement of existing as well as for the development of new networking approaches to keep in step with rising numbers of devices to be connected and increasing amounts of data to be exchanged in RT in the future IIoT and IoT, respectively. The existence of related work in this field speaks for the validity of this hypothesis and hopefully motivates the research community regarding this. The presented HaRTKad approach represents a decentralized alternative to existing solutions but its performance still needs to be determined in large-scale networks.

A comprehensive modeling approach like joint routing and scheduling revealed to be the method of choice for routing and scheduling of RT networks. Prospectively, new scheduling strategies need to be developed that take the benefits of the joint routing and scheduling approach into account and consider complex application constraints as well, because current scheduling strategies are not sufficiently scalable to solve such a comprehensive model for large-scale networks. During the development of these new scheduling strategies, it will be necessary to refine the benchmarks used, such that different approaches can be compared in a meaningful way. However, if the algorithms often need to react to dynamism in the network at runtime the size of the problem instances to be solved becomes an even more serious issue in terms of computation time. New approximation algorithms for scheduling and routing with polynomial runtime can help to reduce computation times.

The management of a wireless mesh management revealed to be able to improve its performance by enabling cross-layer collaboration between the mesh link layer and the application layer already in a small test bed. In future work, larger and more dynamic scenarios with moving participants will be investigated. The presented approach for cross-layer collaboration between IEEE 802.11s and BT is expected to perform even better in a large-scale setup since higher node and path diversity also increase the impact of peer selection. Moreover, the RT capability of wireless mesh networks will be analyzed since communication in the prospective IIoT requires RT capability.

Further, distributed applications in dynamic opportunistic networks with highly dynamic participants have been shown to be both accurate and scalable. However, the dynamics due to mobility imposes tight constraints on the convergence speed of the algorithm. Although simulation results provide promising results, as part of future work it is planned to evaluate other scenarios in a realistic testbed, including an investigation of interference and path-loss effects as well as communication technology-specific impacts.

Bibliography

[1] P. C. Evans and M. Annunziata, "Industrial Internet: Pushing the Boundaries of Minds and Machines", General Electric, Tech. Rep., 2012.

[2] Communication Promoters Group of the Industry-Science Research Alliance, acatech - National Academy of Science and Engineering, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0", Industrie 4.0 Working Group, Tech. Rep., 2013.

[3] A. S. Tanenbaum and M. v. Steen, *Distributed Systems: Principles and Paradigms (2Nd Edition)*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 2016, ISBN: 978-15-302817-5-6.

[4] J. A. Stankovic, "Misconceptions about real-time computing: A serious problem for next-generation systems", *Computer*, vol. 21, no. 10, pp. 10–19, 1988, ISSN: 0018-9162. DOI: `10.1109/2.7053`.

[5] Deutsches Institu für Normung (DIN), *Din 44300*, Beuth-Verlag, Ed., 1985.

[6] P. Danielis, M. Gotzmann, D. Timmermann, T. Bahls, and D. Duchow, "A peer-to-peer-based storage platform for storing session data in internet access networks", in *World Telecommunications Congress (WTC 2010)*, Sep. 2010, pp. 5–10.

[7] R. Brunner, "A performance evaluation of the kad-protocol", Master's thesis, Univ. of Mannheim, Nov. 2006.

[8] P. Maymounkov and D. Mazières, "Kademlia: A peer-to-peer information system based on the xor metric", in *1st International Workshop on Peer-to-Peer Systems*, ser. LNCS, vol. 2429, Springer, 2002, pp. 53–65, ISBN: 3-540-44179-4. [Online]. Available: `http://dl.acm.org/citation.cfm?id=646334.687801`.

[9] P. Danielis, J. Skodzik, V. Altmann, L. Lender, and D. Timmermann, "Dynamic search tolerance at runtime for lookup determinism in the dht-based p2p network kad", in *Consumer Communications & Networking Conference*, IEEE, Jan. 2015, pp. 357–362.

[10] P. Danielis, V. Altmann, J. Skodzik, T. Wegner, A. Körner, and D. Timmermann, "P-donas: A p2p-based domain name system in access networks", *ACM Trans. Internet Technol.*, Jul. 2015.

[11] J. Skodzik, P. Danielis, V. Altmann, and D. Timmermann, "Extensive analysis of a kad-based distributed storage system for session data", in *Symposium on Computers and Communications (ISCC 2013)*, IEEE, Jul. 2013, pp. 489–494. DOI: `10.1109/ISCC.2013.6754994`.

[12] J. Skodzik, P. Danielis, V. Altmann, and D. Timmermann, "Hartkad: A hard real-time kademlia approach", in *11th Consumer Communications & Networking Conference*, IEEE, Jan. 2014, pp. 566–571.

[13] J. Skodzik, V. Altmann, P. Danielis, A. Wall, and D. Timmermann, "A kad prototype for time synchronization in real-time automation scenarios", in *World Telecommunications Congress (WTC 2014)*, Jun. 2014, pp. 1–6.

[14] J. Skodzik, P. Danielis, V. Altmann, and D. Timmermann, "Time synchronization in the dht-based p2p network kad for real-time automation scenarios", in *2nd WoWMoM Workshop on the Internet of Things: Smart Objects and Services (IoT-SoS)*, IEEE, Jun. 2013, pp. 1–6, ISBN: 978-1-4673-5827-9.

[15] Free Software Foundation Inc., *Lwip - a lightweight tcp/ip stack*, Jun. 2015. [Online]. Available: `http://savannah.nongnu.org/projects/lwip`.

[16] B. Konieczek, J. Skodzik, P. Danielis, V. Altmann, M. Rethfeldt, and D. Timmermann, "Hartkad: A p2p-based concept for deterministic communication and its limitations", in *2016 IEEE Symposium on Computers and Communication (ISCC)*, 2016, pp. 1157–1162. DOI: `10.1109/ISCC.2016.7543893`.

[17] G. A. Ditzel and P. Didier, "Time sensitive network (tsn) protocols and use in ethernet/ip systems", in *2015 ODVA Industry Conference & 17th Annual Meeting*, 2015.

[18] Anybus News, *Industrial Ethernet is now bigger than fieldbuses*, 2018. [Online]. Available: `https://www.anybus.com/about-us/news/2018/02/16/industrial-ethernet-is-now-bigger-than-fieldbuses`.

[19] N. G. Nayak, F. Dürr, and K. Rothermel, "Time-sensitive software-defined network (tssdn) for real-time applications", in *Proceedings of the 24th International Conference on Real-Time Networks and Systems*, ACM, 2016, pp. 193–202.

[20] N. Feamster, J. Rexford, and E. Zegura, "The road to sdn: An intellectual history of programmable networks", *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, 2014.

[21] Open Networking Foundation. (2017). Software-defined networking (sdn) definition, [Online]. Available: `https://www.opennetworking.org/sdn-definition/`.

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

[23] SDxCentral. (2017). What's software-defined networking (sdn)?, [Online]. Available: `https://www.sdxcentral.com/sdn/definitions/what-the-definition-of-software-defined-networking-sdn/`.

[24] ——, (2017). What are sdn northbound apis?, [Online]. Available: `https://www.sdxcentral.com/sdn/definitions/north-bound-interfaces-api/`.

[25] J. W. Guck and W. Kellerer, "Achieving end-to-end real-time quality of service with software defined networking", in *IEEE Intl. Conf. on Cloud Networking (CloudNet)*, Oct. 2014, pp. 70–76.

[26] H. Puttnies, B. Konieczek, J. Heller, D. Timmermann, and P. Danielis, "Algorithmic approach to estimate variant software latencies for latency-sensitive networking", in *Information Technology, Electronics and Mobile Communication Conference (IEMCON), 2016 IEEE 7th Annual*, IEEE, 2016, pp. 1–7.

[27] H. Puttnies, D. Timmermann, and P. Danielis, "An approach for precise, scalable, and platform independent clock synchronization", in *Consumer Communications & Networking Conference (CCNC), 2017 14th IEEE Annual*, IEEE, 2017, pp. 461–466.

[28] J. Farkas, "Introduction to ieee 802.1: Focus on the time-sensitive networking task group", TSN Task Group, 2017.

[29] TTTech Computertechnik AG, "Ieee tsn (time-sensitive networking): A deterministic ethernet standard", in. 2015.

[30] IEEE Standards Association. (2015). Ieee p802.1qbu/d3.0, [Online]. Available: `http://www.ieee802.org/1/pages/802.1bu.html`.

[31] IET Task Force. (2015). Ieee 802.3br interspersing express traffic (iet) task force (tf), [Online]. Available: `http://www.ieee802.org/3/br/Baseline/8023-IET-TF-1405_Winkel-iet-Baseline-r4.pdf`.

[32] IEEE Standard for Ethernet. (2015), [Online]. Available: `https://standards.ieee.org/findstds/standard/802.3-2015.html`.

[33] IEEE Standard for Local and metropolitan area networks - Bridges and Bridged Networks. (2014), [Online]. Available: `http://standards.ieee.org/getieee802/download/802-1Q-2014.pdf`.

[34] E. Schweissguth, P. Danielis, C. Niemann, and D. Timmermann, "Application-aware industrial ethernet based on an sdn-supported tdma approach", in *2016 IEEE World Conference on Factory Communication Systems (WFCS)*, 2016, pp. 1–8. DOI: `10.1109/WFCS.2016.7496496`.

[35] (). Pox on github, [Online]. Available: `https://github.com/noxrepo/pox`.

[36] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A survey", *Computer networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[37] IEEE Standard for Information Technology, *Telecommunications and information exchange between systems Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, 2012.

[38] R. C. Carrano, L. C. Magalhães, D. C. M. Saade, and C. V. Albuquerque, "Ieee 802.11 s multihop mac: A tutorial", *IEEE Communications Surveys & Tutorials*, vol. 13, no. 1, pp. 52–67, 2011.

[39] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, “Ieee 802.11 s: The wlan mesh standard”, *IEEE Wireless Communications*, vol. 17, no. 1, 2010.

[40] M. Rethfeldt, P. Danielis, G. Moritz, B. Konieczek, and D. Timmermann, “Design and development of a management solution for wireless mesh networks based on ieee 802.11 s”, in *Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on*, IEEE, 2015, pp. 902–905.

[41] M. Rethfeldt, A. Wall, P. Danielis, B. Konieczek, and D. Timmermann, “Akademesh: Software-defined overlay adaptation for the management of ieee 802.11 s networks”, in *Consumer Communications & Networking Conference (CCNC), 2016 13th IEEE Annual*, IEEE, 2016, pp. 477–482.

[42] B. Cohen, “Incentives build robustness in bittorrent”, in *Workshop on Economics of Peer-to-Peer systems*, vol. 6, 2003, pp. 68–72.

[43] V. Sevani, B. Raman, and P. Joshi, “Implementation-based evaluation of a full-fledged multihop tdma-mac for wifi mesh networks”, *IEEE Transactions on Mobile Computing*, vol. 13, no. 2, pp. 392–406, 2014.

[44] Index, Cisco Visual Networking, “Cisco visual networking index: Global mobile data traffic forecast update, 2014–2019”, *Tech. Rep*, 2015.

[45] Ó. Helgason, E. A. Yavuz, S. Kouyoumdjieva, L. Pajevic, and G. Karlsson, “A mobile peer-to-peer system for opportunistic content-centric networking”, in *Proc. ACM SIGCOMM MobiHeld workshop*, New Delhi, India, 2010.

[46] S. T. Kouyoumdjieva, Ó. R. Helgason, and G. Karlsson, *CRAWDAD data set kth/walkers (v. 2014-05-05)*, Downloaded from http://crawdad.org/kth/walkers/, May 2014.

[47] *Legion Studio*, `http://www.legion.com/`.

[48] Ó. Helgason, S. T. Kouyoumdjieva, and G. Karlsson, “Opportunistic communication and human mobility”, *Mobile Computing, IEEE Transactions on*, vol. 13, no. 7, pp. 1597–1610, 2014, ISSN: 1536-1233. DOI: `10.1109/TMC.2013.160`.

[49] M. S. Desta, E. Hyytiä, J. Ott, and J. Kangasharju, “Characterizing content sharing properties for mobile users in open city squares”, in *Wireless On-demand Network Systems and Services (WONS), 2013 10th Annual Conference on*, 2013, pp. 147–154. DOI: `10.1109/WONS.2013.6578340`.

[50] Y. Benkaouz, R. Guerraoui, M. Erradi, and F. Huc, “A distributed polling with probabilistic privacy”, in *Reliable Distributed Systems (SRDS), 2013 IEEE 32nd International Symposium on*, 2013, pp. 41–50.

[51] A. Kuznetsov, *D-gap compression*, 2002. [Online]. Available: `http://bmagic.sourceforge.net/dGap.html`.

[52] Ó. R. Helgason and K. V. Jónsson, “Opportunistic networking in OMNeT++”, in *Proc. SIMUTools, OMNeT++ workshop*, Marseille, France, 2008.

[53] Ó. Helgason, S. T. Kouyoumdjieva, and G. Karlsson, “Opportunistic communication and human mobility”, *IEEE Transactions on Mobile Computing*, vol. 13, no. 7, pp. 1597–1610, 2014, ISSN: 1536-1233. DOI: `10.1109/TMC.2013.160`.

[54] M. Rethfeldt, H. Raddatz, B. Beichler, B. Konieczek, D. Timmermann, C. Haubelt, and P. Danielis, “Vipmesh: A virtual prototyping framework for ieee 802.11 s wireless mesh networks”, in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2016 IEEE 12th International Conference on*, IEEE, 2016, pp. 1–7.

Papers

Paper A

Survey on Real-Time Communication Via Ethernet in Industrial Automation Environments

Peter Danielis, Jan Skodzik, Vlado Altmann, Eike B. Schweissguth, Frank Golatowski, Dirk Timmermann, and Joerg Schacht

In Proceedings of the 19th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA), pp. 1-8,
`https://doi.org/10.1109/ETFA.2014.7005074`, ISBN: 978-1-4799-4846-8,
September 2014

Survey on Real-Time Communication Via Ethernet in Industrial Automation Environments

Peter Danielis, Jan Skodzik, Vlado Altmann, Eike B. Schweissguth, Frank Golatowski,
and Dirk Timmermann
University of Rostock, 18051 Rostock, Germany
Email: {peter.danielis;jan.skodzik}@uni-rostock.de

Joerg Schacht
Max Planck Institute of Plasma Physics, 17489 Greifswald, Germany

Abstract

For companies in the automation industry, the development of real-time Ethernet to connect devices is of high economic interest to replace conventional fieldbus systems. Therefore, many approaches for adapting Ethernet to real-time requirements come from industrial applications. This is a challenging task as the original Ethernet standard IEEE 802.3 was not designed for real-time data transmission. Likewise, protocols basing on Ethernet like TCP, UDP, and IP do typically not consider real-time requirements. Hence, adaptations on several OSI layers become necessary to make the industrial system meet hard real-time requirements. For this purpose, a multitude of real-time capable Industrial Ethernet systems has been developed, which solve the problems of standard Ethernet- and TCP/IP- or UDP/IP-based communication in a variety of ways. This paper gives a summary of different Industrial Ethernet protocols for the real-time data transmission via Ethernet in automation environments. Advantages and disadvantages of these protocols are analyzed with regard to their sustainability in terms of their real-time capability, reliability, scalability, self-configuration of the network, and hardware requirements. Against the background of connected devices tremendously growing in number and computational power in the prospective "Industrial Internet", consequences for future developments are drawn.

1 Introduction

After the industry has already undergone three revolutions in the form of mechanization, electrification, and informatization, as fourth industrial revolution, the Internet of Things and Services is predicted to find its way into the factory. For this development, e.g., in Germany the term "Industry 4.0" has been coined [1]. In the most general sense, globally a networked and real-time (RT) capable industrial production is aspired. The

US-American company General Electric (GE) recently initiated a comprehensive research initiative called "Industrial Internet" [2]. Thereby, not only the industrial production but also the whole industrial infrastructure shall be intelligently networked. GE forecasts for the future that there will be more intelligent devices, which have to be connected to interact with each other dynamically.

As part of these efforts, companies aim at connecting their devices, storage systems, and supplies as cyber-physical systems (CPS) prospectively. In that way, intelligent devices, storage systems, and supplies shall be created in the industrial production, which exchange data in a self-organizing way, trigger actions, and control each other. CPS are systems with embedded software as part of, e.g., manufacturing facilities but can also comprise buildings and devices, which collect physical data by means of sensors and influence physical processes with actors. CPS are networked among each other with local digital communication systems but also with global networks. By connecting embedded systems with global networks, on the one hand numerous applications for all parts of daily life and novel business opportunities emerge. On the other hand, this poses the challenge of bringing together the features of embedded systems like RT requirements with the openness of the Internet [3].

Finally, the vision is the so-called Smart Factory with a novel production logic: The products are intelligent and can be identified clearly, constantly located, and are aware of their current state. These embedded production systems shall be interconnected with economic processes vertically and combined to a distributed RT capable network horizontally. To reach this vision, the production systems must be flexible and adaptable. Therefore, automation structures are necessary to manage the high complexity arising from the increasing number of devices. Furthermore, the control of a network consisting of thousands of devices is a technical challenge requiring tools and technologies to be able to meet this challenge. As soon as the number of devices increases from 100 to 1,000 or even 10,000, the device networking technology must be ready for this magnitude. Consequently, e.g., the "Industry 4.0" Working Group urges to meet the requirements of guaranteed latencies, i.e., RT capability and high resilience for the desired massive interconnection to ensure the frictionless functionality of the respective applications [1].

To conclude, devices have to be interconnected in industrial facilities to communicate with each other and prospectively, their number will strongly increase in the described Internet of Things and Services. Therefore, this paper first elaborates on the current development from fieldbuses to Ethernet as *device networking technology for devices in the Internet of Things of Services* in Section 2. In Section 3, established RT capable Ethernet systems are presented and compared. In the following Section 4, against the background of connected devices tremendously growing in number and computational power, consequences for future developments are drawn and current developments are sketched. Finally, the paper concludes in Section 6.

2 Current Development: From Fieldbuses to Ethernet

For the RT capable device interconnection in industrial environments, originally various fieldbus solutions have been used. However, fieldbuses are subject to severe restrictions concerning the number of devices that can be networked thereby limiting the scalability and resilience. Furthermore, interoperability between the various solutions is not provided. Therefore, the networking by means of the wide-spread Ethernet technology currently prevails against fieldbuses [4]. The application of Ethernet to connect field devices offers substantial advantages compared to fieldbuses as Ethernet enables the consistent integration on all levels of a company. Thereby, Ethernet solutions allow for a total vertical and horizontal integration of an automation system from the field level up to company level [5, 6], which is of decisive importance to realize the vision of an Industrial Internet. There are many challenges, e.g., addressing the data representation, which have to be faced, to especially realize a vertical integration of the system as field devices can now communicate with PCs in an office without any gateway between them. This is one future implementation detail, which has to be considered, to properly represent and process data from the lowest to the highest level in an automation scenario [7].

Via Ethernet connections between field devices, time-critical process data, which fulfills control tasks, is to be transmitted meanwhile non-time-critical IT data is sent to different IT services in the company [8]. In the case of this type of communication, status information can be read out or field devices can be controlled remotely. For the transmission of IT data, standard protocols like TCP/IP or UDP/IP can be used whereby time-critical process data may require the application of special protocols. The entire communication via Ethernet takes place on a common hardware base, which has been widely standardized by IEEE 802.3 and offers different plug connections and communication media, which can be adjusted to the specific purpose. Another advantage of Ethernet is the high performance compared to conventional fieldbuses, which are evolving into weak spots between powerful computer systems.

However, standard Ethernet as used today in many areas comprises various mechanisms, which prevent a deterministic data transmission and therefore the application in environments with RT requirements. The first problem of standard Ethernet is its application of the CSMA/CD access method. Due to possible collisions, the data transmission may be interrupted or can only take place at a later undetermined point in time. By using full-duplex switched Ethernet, the problem can be solved but new problems arise. In switches, data is buffered, i.e., frames are put into a queue on the switches, which leads to additional delay or even packet loss under specific traffic load conditions. If, e.g., several network devices send much data to the same destination via one switch buffer overflow and packet loss must be expected, which contradicts a deterministic data transmission. On the IP layer above Ethernet, the problem of non-static routes exists so that the way and transit time of a packet are not precisely predictable. Likewise, the choice of the transport protocol for meeting high RT requirements is challenging to be able to transport larger amounts of data than solely process data in future applications while ensuring deterministic predictable transmission times. This could become more important in the future, which, e.g., the current works of the IEEE Time-Sensitive Networking (TSN) Task Group show aiming at allowing for time-synchronized low latency streaming services through 802 networks [9]. As automotive engineers have shown interest in these works for, e.g., distributing streams in vehicular communication systems [10], the works of the IEEE TSN task group could become prevalent in industrial automation as well. While UDP can be basically considered as RT capable due to sending single independent packets, it does not provide the labeling of related data or data order, which is indispensable for the reliable transmission of larger amounts of data. Contrary, TCP has been designed for the transmission of large amounts of data and therefore labels related data and their order. However, due to the variable transmission speed, which is caused by mechanisms for overload control and error correction, TCP is fundamentally not RT capable.

A multitude of RT capable Industrial Ethernet (IE) systems has already been developed, which remedy the deficiencies of standard Ethernet and TCP/IP or UDP/IP-based communication in a variety of ways [11]. "Industrial" refers to the compatibility of the solutions with rough industrial environments [12]. Therefore, certifications are carried out to prove the compliance with the regulations. As standard Ethernet uses the CSMA/CD mechanism to control the media access, neither the arrival of an Ethernet frame nor its delivery time can be guaranteed. However, to be used in machine halls, IE solutions must provide guarantees in this regard. Control systems have to be implemented as RT systems so that data is transmitted within fixed time limits. Neither switched nor full-duplex Ethernet can ensure fixed time limits and moreover, it cannot restrict the jitter of Ethernet frames. For multimedia applications, this problem has been solved by prioritizing the data flow and by using virtual LAN standards [13, 14]. Thereby, the jitter is reduced and delivery times of 10 ms in case of highly prioritized data traffic is reached. However, these are only statistical assessments. For ensuring lower deterministic delivery times, either the Ethernet protocol itself has to be modified and/or the network components/devices have to take measures to manage the communication. RT for Ethernet can, e.g., be achieved by applying a token-passing procedure as done in [15]. However, the monitoring of the token and the participants is required. If a participant fails, the previous station usually sends the message in the opposite direction as the communication otherwise fails completely. Moreover, a message has to be passed to all participants on the token ring, which can lead to high latency in case of a very high number of participants. Therefore, in most IE systems, a time division multiplex method is applied whereby each device is allowed to communicate in a time slot. This requires a common time base for all devices, i.e., a synchronization among all devices is required.

This paper analyzes the advantages and disadvantages of RT capable IE systems with regard to their sustainability in terms of

- RT capability: Which performance and thus what kind of RT can be achieved in the best case?

- Reliability: Does the network contain a single point of failure (SPoF)?
- Scalability: How many devices can exchange data in RT? Prospectively, there will probably be several thousands of devices to be connected [1, 2].
- Self-configuration: Does the system show self-configuration features like dynamical adaptation to changes of the network topology or does it have to be statically/-manually configured?
- Hardware requirements: Is special Ethernet hardware needed?

3 RT capable Industrial Ethernet Systems

A system is called RT capable if the response of a system to a request does not exceed a given time limit [16], i.e., such RT systems have the distinction of keeping time conditions given by applications. That is, the correctness of a RT system not only depends on the correct computation result but also on the time of the result computation [17]. Depending on to what extent the time conditions are mandatory the type of RT is referred to as soft or hard. Soft RT means that the application allows for keeping the time conditions substantially, i.e., time limits may be exceeded slightly without damage incurring. For the definition of hard RT conditions, the following requirement called punctuality is given in Equation 1.

$$A \equiv r + \Delta e \leq d \tag{1}$$

r denotes the point in time, at which a task starts. The task execution takes a time span of Δe and the task must be attended at the point in time d. Hard RT systems are characterized by definitely keeping the time condition A under the boundary condition B so that the condition given by Equation 2 applies:

$$P(A|B) = 1 \tag{2}$$

Depending on the specific task, B indicates that within the time span Δe, elapsing from r to d, neither technical faults occur nor more important tasks have to be attended.

For the precise description of the RT capabilities of IE systems, three classes can be defined depending on the cycle times of the IE systems [18, 19]:

- Class 1: soft RT: scalable cycle time, approx. 100 ms.
- Class 2: hard RT: cycle time 1 to 10 ms.
- Class 3: isochronous RT: cycle time 250 μs to 1 ms; jitter less than 1 μs.

This classification is used in the following and each IE system is classified according to its best RT class achievable. Moreover, an overview of the best delivery time achievable by each system is given.

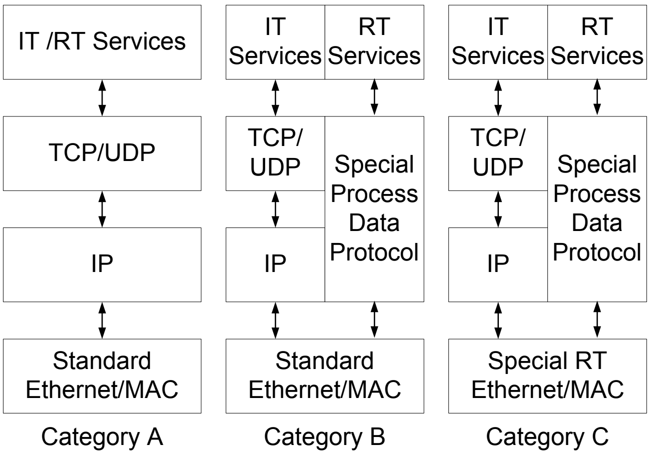


Figure 1: Categorization of IE systems in terms of software and hardware needs for device implementations [8, 20].

3.1 Established IE Systems

This section gives an overview of established IE systems. In addition to the classification in terms of RT capability, each of the systems is categorized according to software and hardware needs for device implementations (see Figure 1) [8, 20]:

- Category A: Both IT and RT services are completely TCP/UDP/IP-based and use standard Ethernet controllers and switches.
- Category B: Both IT and RT services use ordinary Ethernet controllers and switches but a special process data protocol is introduced on top of Ethernet.
- Category C: RT services require a dedicated process data protocol like category B and additionally require special RT Ethernet controllers and switches.

3.1.1 Modbus-TCP

Modbus-TCP is solely designed for soft RT applications and therefore falls into class 1. It is located on the application layer on top of TCP/IP and completely bases on standard Ethernet components (if used on top of Ethernet) making it one of the few category A approaches [20]. It has a well known TCP port 502 to transmit data and can therefore be controlled remotely [21]. As it is located on the application layer, the protocol can be applied to all device types for communication and is very easy to configure, i.e., shows a high degree of self-configuration. It is implemented as client-server architecture whereby each device may become client or server, which contributes to the high reliability of Modbus-TCP. The server processes requests of clients and confirms them with a success or error message. The number of devices connected by Modbus-TCP is practically only limited by the computational power of the devices themselves.

3.1.2 Ethernet Powerlink

Ethernet Powerlink basically allows RT communication with standard Ethernet hardware as the protocol can be completely implemented in software (category B) [22]. However, for applications putting high requirements on cycle times of RT communication, i.e., isochronous RT (class 3), an implementation of the protocol in hardware is necessary (category C) [8]. The Powerlink protocol is located between Ethernet and IP. The developed time-slotted access method to Ethernet connections bases on a master-slave concept called Slot Communication Network Management, which provides transmission capacities for cyclic process data as well as acyclic service and control data. The master, called Managing Node by Powerlink, gives permission to the slaves to send data sequentially during a cycle. Hence, the master is decisive for Powerlink's successful operation and must not fail as otherwise, the total system is put out of operation.

The scalability in terms of connected devices is limited as Powerlink uses own 8 bit addresses. The addresses have to be manually configured but Powerlink is hot-plug capable, i.e., devices can be integrated or removed during runtime therefore allowing for some degree of self-configuration. On the application layer, Powerlink uses the CANopen standard for RT capable data transmission, which can solely be used for the transmission of process data rather than large data amounts.

3.1.3 EtherCAT

EtherCAT is an IE system, which bases on the master-slave principle and applies a procedure for the processing of cyclic process data in field devices (slaves) [23, 8].

For the communication, a process image is created in the master, which represents the state of various in- and outputs of the overall system comprising several slaves. To change the state of specific outputs of a slave, the respective part of the process image together with a change command has to be dispatched. Slaves themselves can send parts of the process image to the master during the cyclic data exchange to update state information of their inputs. The assignment of these parts of the process image to in- and outputs of the single slaves takes places via logical addresses, which are translated to physical addresses of the particular devices in the EtherCAT Slave Controller (ESC) [23, 8, 24]. Parts of the process image and the respective command to change outputs are dispatched either directly in Ethernet frames (used for achieving isochronous RT within a subnet) or as UDP payload (not RT capable for sending data from another subnet). These EtherCAT frames are cyclically sent from the master and pass the slaves sequentially on a ring structure. Contrary to common procedures of standard Ethernet controllers, which buffer and process an incoming frame and send a new one, the processing takes place completely in hardware while the frame passes the ESC (category C) [23].

EtherCAT provides a high-performant system for RT data transmission via Ethernet. By means of the fast data processing in the ESC and due to low overhead, cycle times far below 1 ms are possible (class 3). EtherCAT is intended to be easily diagnosed and configured and therefore provides configuration tools, which are able to depict the network topology and automatically configure the devices in the network. However, a master must be permanently available, which undertakes the administrative tasks and the storage of all data and addresses available in the network. The scalability is limited due to the logical process image. The advantage of the low overhead by aggregating data to various recipients would reduce substantially as soon as the network would be used for sending larger amounts of data than process data—if possible at all.

3.1.4 TCnet

TCnet developed by Toshiba extends the original access method of IEEE 802.3 to Ethernet connections and introduces four communication classes with different priorities.

The four communication classes are (sorted by transmission priority):

- Cyclic data with high timing requirements (class 3)
- Cyclic data with medium timing requirements
- Acyclic data
- Cyclic data with low timing requirements

The cyclic data transmission is used by RT application and can only take place inside one subnetwork as data is directly encapsulated in an Ethernet frame.

To enable the new access method, each TCnet participant needs special TCnet Ethernet controllers (category C). Prior to commissioning, a station number for each participant has to be configured manually, which determines the transmission sequence for the cyclic data transmission [25, 26]. Another principle used by TCnet is a common memory of the TCnet stations participating in the RT communication. Each station contains an own copy of the common memory and can therefore access all process data at any time. Thereby, the RT communication is solely intended for process data being reflected in the memory size of 256 KByte defined by Toshiba. Such a memory size is no longer feasible if larger amounts of data have to be transmitted.

3.1.5 TTEthernet

TTEthernet is a RT capable IE system, which can be basically combined with standard Ethernet systems and thereby enables a wide variety of applications.

TTEthernet defines three message types (time triggered, rate constrained, best effort) whereby the time triggered message type is designed for isochronous RT applications (class 3) [27]. To be able to send data of the stated message types, special switches are required supporting the TTEthernet protocol (category C). To dispatch TT packets at the right point in time and to block the transmission channels during this timeframe for other message types, a common time base of the switches and connected devices is indispensable. It is introduced by one or more master devices, which have to be configured manually, by means of the synchronization of distributed clocks.

TTEthernet provides a deterministic transmission method on the two lowest OSI layers. Therefore, for higher layers the transmission by TTEthernet is completely transparent and there are no specifications, which kind of data is transmitted. Basically, a RT transmission mechanism for larger amounts of data could be integrated here.

3.1.6 CC-Link IE Field

CC-Link IE Field from Mitsubishi Electric bases on full-duplex Gigabit Ethernet so that the topology is almost arbitrary and achieves hard RT (class 2 [20]). The addressing of field devices takes place by station numbers ranging from 1 to 254 so that the number of devices is limited [8]. Similar to other IE systems, CC-Link IE Field uses a master-slave principle. Thereby, the master is responsible for the initialization of the total network and controls the data transmission. It stores the state of all in- and outputs of the devices in the network as well. The information memory of the master is limited to 32,768 bits and 16,384 words and represents a summary of all device memories in the network. The data transmission takes place cyclically by means of a token passing mechanism. Starting with the master, the token is passed in the network, which allows the current token owner to send data. For realizing CC-Field IE Field either special hardware in the slave (Mitsubishi CP220 Chip) or an implementation of the Seamless Message Protocols is required (category C) [28, 29]. As stated in [20], for third parties the implementation of CC-Link IE Field is challenging leading to no visible introduction outside Mitsubishi.

3.1.7 Profinet

The communication in Profinet takes place cyclically and is divided into several phases [30]. Each cycle starts with the isochronous phase, in which Isochronous RT (IRT) frames are transmitted (class 3). The transmission of IRT frames is already configured during the installation of the network. By means of synchronized clocks, in each device the point in time is precisely scheduled when a IRT frame may be sent. The synchronization is carried out by a master. In spite of data transmission in Ethernet frames, the addressing does not take place by MAC addresses but frames are forwarded through switches on a fixed route depending on the transmission time. Therefore, special Profinet switches are required (category C). After the isochronous phase, another RT phase follows and finally, a phase for non-time critical data transmitted via UDP or TCP is provided [26]. Due to [20], the crucial issue of Profinet IRT is the complex system planning but still Profinet gained a noticeably high market share (14.5 % estimated for 2015) due to Siemens encouraging and supporting its development.

3.1.8 EtherNet/IP

Ethernet/IP bases on the Common Industrial Protocol (CIP), which is located on top of TCP/IP and UDP/IP (category A) [31]. For the RT transmission of process data, UDP/IP is used whereby a direct communication between all devices is possible. Ethernet/IP can be operated with all protocols on the application layer and a limitation of the number of devices is basically not given. However, in practice there is an obvious limitation if isochronous RT is to be achieved. Moreover, if isochronous RT has to be reached (class 3), the time synchronization has to be implemented in hardware by means of special switches with built-in IEEE 1588 timestamp support (category C) as otherwise the stack performance is not sufficient. To complicate matters, routers must have multicast/broadcast control features available and there is no standard to implement or configure these features [20].

3.1.9 SERCOS III

SERCOS III organizes devices as double ring structure with hardware redundancy or as line structure without hardware redundancy [32]. Per ring/line, a maximum of 511 devices are permitted thereby limiting the scalability. SERCOS III preserves its communication ability even in the case of an error like cable break or node failures and new devices can be integrated at runtime so SERCOS III provides a high degree of self-configuration and flexibility. The communication in SERCOS III is grounded on a time-slot method with cyclic telegram transmission on the basis of a master-slave principle. A central master sends so-called Master Data Telegrams to slaves, which can communicate with each other directly as well (Cross Communication and Controller-to-Controller communication profile). The telegrams base on standard Ethernet frames and transport only small amounts of process data. The SERCOS III master can be realized with special hardware to achieve isochronous RT (category C) or alternatively be completely implemented in software (category B, SERCOS III SoftMaster).

3.1.10 Comparison of Established IE Systems

All investigated IE systems show similar basic principles, which are solely implemented in different ways. Actually, several solutions apply a shared memory and most systems require a master (information not specified for TCnet) or a comparable management system, which controls the communication or have to be configured manually. The manual configuration effort, which has to be expended if devices have to be integrated or changed, differs according to the IE system whereby basically the effort is intended to be kept minimal for achieving high flexibility at runtime.

Most isochronous RT capable solutions have in common that new devices have to be recognized by the master to adapt the communication mechanism and assign time slots or a polling or token procedure to new devices. The master as central instance represents a SPoF and bottleneck thereby limiting the system reliability and scalability. A completely self-organizing network, in which devices act autonomously, does not yet exist. Moreover, all realizations require dedicated and expensive hardware to realize isochronous RT behavior.

Another similarity of the systems is the optimization to process data, which is especially demonstrated by the small amounts of data exchanged in RT. None of the IE systems provides for a transport protocol to be able to transport larger amounts of data, e.g., streams in vehicular communication systems [10] while ensuring deterministic predictable

Table 1: Comparison of established IE systems and current developments

IE system	Del. Time [ms]	Class/ Cat.	Rel.: Contains SPoF?	Scal.: Max. devices	Self-conf. features?	Special hardware requirements?	Market share [%] [33]
Modbus-TCP	1-15	1/A	no	Unltd. [8]	yes	no	6.4
Ethernet Powerlink	0.4	3/C	yes	4	yes	optional	4.2
EtherCAT	0.15	3/C	yes	180	yes	yes (EtherCAT Slave) Controller)	3.1
TCnet	2	2/C	n/s	24	no	yes (TCnet controller)	n/a
TTEthernet	n/a	2/C	yes	n/s	no	yes (TTEthernet switches)	n/a
CC-Link IE Field	1.6	2/C	yes	254 [8]	no	optional	0
Profinet	1	3/C	yes	60	no	yes (Profinet switches)	14.5
EtherNet/IP	0.13	3/C	no	90	no	yes (IEEE 1588 switches)	13.9
SERCOS III	0.0398	3/C	yes	9	yes	optional	2.1
DRTP	5-10	3/B	yes	n/s	yes	no	0
DARIEP	0.1 - 0.3	3/C	yes	n/s	no	yes (FPGA synchr.) slave)	0
HaRTKad	0.7	2-3/A	no	n/s	yes	no	0

transmission times. TTEthernet is one exception as it aims at an RT Ethernet solution, which does not restrict the way of the data transmission on upper layers, but does not specify a protocol either.

A comparison (together with current developments) is apparent in Table 1. If available the estimated market share for 2015 of the respective IE system is given as indicator for the system's market penetration. The maximum number of devices is taken from [11] if not specified otherwise and represents the case of minimum delivery time (only RT). Some facts were not available (n/a) or are not specified in the respective descriptions (n/s).

4 Towards the Industrial Internet of Things: Going beyond master-slave patterns and special hardware

To summarize, among the currently established RT Ethernet technologies there is no solution without master component and special hardware. Moreover, none of the existing systems allows for a reliable RT transmission of increasing amounts of data. These issues will become more relevant in the future. As mentioned in [1, 2], the future for the industry will be more intelligent devices to be connected, which can act more dynamically. Facilities as one main area of application may consist of several thousands of devices still requiring RT behavior. For instance, the high flexibility as a rising challenge cannot be ensured in hierarchical and centralized systems due to their highly static behavior. So we think, the existing solutions will not fulfill the future challenges in terms of reliability, scalability, and flexibility as they are right now. They either need revision or new solutions will have to be developed. For example, decentralized distributed systems could help to solve these issues [34].

4.1 Current Developments

Consequently, there are already some developments targeting the weaknesses of established IE solutions.

(1) Distributed Real-Time Protocols for Industrial Control Systems (DRTP): Schmidt et al. [35] propose a RT Ethernet solution (class 3), which can act in a distributed manner and dynamically changes the bandwidth allocation to the shared Ethernet medium. The concept bases on two additional proprietary layers on top of the Ethernet layer to manage the media access and a master, which synchronizes the slaves by means of the IEEE 1588 synchronization protocol (category B). The result is an TDMA-approach using time slots. There is no statement about a high number of attendees and its applicability for large scale networks. Furthermore, as TCP/UDP and IP are not supported the total vertical and horizontal is not possible without further effort. The authors' work sounds promising; especially, the dynamic bandwidth allocation is an interesting self-organization feature. However, many implementation issues are subject to future work and still a central synchronization instance is required.

(2) Design and application of a RT industrial Ethernet protocol (DARIEP): In [36], a RT industrial Ethernet protocol is developed adopting a master-slave pattern (class 3). The master is developed under Linux using Real Time Application Interface and coordinates the RT data exchange with the nodes by precise cyclic timing. Moreover, the slaves base on universal FPGA and ARM chips so special hardware is necessary (category C). The authors' approach seems to be a high-performant alternative for current IE system at the expense of the need for dedicated hardware.

(3) HaRTKad: A Hard Real-Time Kademlia Approach (RTKad): Skodzik et al. [37] sketches a completely decentralized approach to realize a fully decentralized Kad network meeting at least hard RT constraints for connecting devices in automation scenarios

(class 2-3). They renounce using a master but let the peers synchronize themselves by a decentralized algorithm, which makes the network comparable to Modbus-TCP in terms of high reliability [38, 39]. Each peer is assigned a time slot depending on its hash value and can communicate during this time slot in RT. A RT Kad prototype has been developed, which runs on standard Ethernet hardware on top of UDP (category A) [37]. By applying the P2P paradigm as device connection technology, basically the SPoF in terms of synchronization and communication can be avoided. Together with the high capability of self-organization and the restriction to standard hardware, the approach is appealing.

For a complete comparison of current developments with established IE systems, see Table 1.

5 Conclusion

It becomes apparent that none of the established IE systems meets all requirements concerning high reliable, scalability, flexibility in terms of self-configuration and cost-effective standard Ethernet hardware. No currently established solution achieves isochronous RT data transmission without special hardware and no SPoF at the same time. The protocols on application layer introduced or applied by the established IE systems are optimized to the transmission of process data and do not allow for the reliable RT transmission of increasing amounts of data. Only TTEthernet offers degrees of freedom in terms of a protocol for the deterministic transmission of any data but does not specify a protocol either. Current developments achieve isochronous RT but all except one require special hardware and contain a SPoF limiting their reliability and scalability.

Finally, the authors conclude that there is a need for the advancement of existing and for the development of new IE approaches to keep in step with a rising numbers of devices to be connected and increasing amounts of data to be exchanged in RT in the future Industrial Internet and Internet of Things and Services, respectively. The existence of related work in this field speaks for the validity of this hypothesis and the authors hope to have motivated the research community regarding this.

Acknowledgment

The authors would like to thank Kerstin Thurow, professor for Automation Technology / Life Science Automation at the University of Rostock, Germany, and Thilo Sauter, director of the Institute for Integrated Sensor Systems, Austrian Academy of Sciences, for their advice and assistance in the preparation of this contribution.

This work is partly granted by the Research Fund Mecklenburg-West Pomerania, Germany, as well as the European Social Fund.

6 References

[1] Communication Promoters Group of the Industry-Science Research Alliance, acatech - National Academy of Science and Engineering, "Recommendations for implementing the strategic initiative INDUSTRIE 4.0", Industrie 4.0 Working Group, Tech. Rep., 2013.

[2] P. C. Evans and M. Annunziata, "Industrial Internet: Pushing the Boundaries of Minds and Machines", General Electric, Tech. Rep., 2012.

[3] acatech - National Academy of Science and Engineering, "Cyber-physical systems. driving force for innovation in mobility, health, energy and production.", Tech. Rep., 2011. [Online]. Available: `http://www.acatech.de/de/publikationen/stellungnahmen/kooperationen/detail/artikel/cyber-physical-systems-innovationsmotor-fuer-mobilitaet-gesundheit-energie-und-produktion.html`.

[4] Panel Building & System Integration, *Ethernet adoption in process automation to double by 2016*, 2013. [Online]. Available: `http://www.pbsionthenet.net/article/58823/Ethernet-adoption-in-process-automation-to-double-by-2016.aspx`.

[5] T. Sauter, "Integration aspects in automation - a technology survey", in *10th IEEE Conference on Emerging Technologies and Factory Automation*, 2005, pp. 255–263.

[6] ——, "The three generations of field-level networks 2014—evolution and compatibility issues", *Industrial Electronics, IEEE Transactions on*, vol. 57, no. 11, pp. 3585–3595, 2010, ISSN: 0278-0046. DOI: `10.1109/TIE.2010.2062473`.

[7] T. Sauter and M. Lobashov, "How to access factory floor information using internet technologies and gateways", *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 699–712, 2011, ISSN: 1551-3203. DOI: `10.1109/TII.2011.2166788`.

[8] F. Klasen, V. Oestreich, and M. Volz, *Industrial Communication with Fieldbus and Ethernet.* VDE Verlag GmbH, 2011.

[9] IEEE 802 LAN/MAN Standards Committee, *Time-sensitive networking task group*, 2014. [Online]. Available: `http://www.ieee802.org/1/pages/tsn.html`.

[10] T. Steinbach, H.-T. Lim, F. Korf, T. Schmidt, D. Herrscher, and A. Wolisz, "Tomorrow's in-car interconnect? a competitive evaluation of ieee 802.1 avb and time-triggered ethernet (as6802)", in *2012 IEEE Vehicular Technology Conference (VTC Fall)*, 2012, pp. 1–5. DOI: `10.1109/VTCFall.2012.6398932`.

[11] M. Felser, "Real time ethernet: Standardization and implementations", in *IEEE International Symposium on Industrial Electronics*, 2010.

[12] B. Wilamowski, *Industrial Communication Systems (The Industrial Electronics Handbook).* CRC Press, 2011.

[13] *Ieee 802.1d edition 2004 ieee standard for local and metropolitan area networks - media access control (mac) bridges*, New York: Institute of Electrical and Electronics Engineers, 2004.

[14] *Ieee 802.1q edition 2005 ieee standard for local and metropolitan area networks - virtual bridged local area networks*, Institute of Electrical and Electronics Engineers, 2005.

[15] R. Moraes, F. B. Carreiro, P. Bartolomeu, V. Silva, J. A. Fonseca, and F. Vasques, “Enforcing the timing behavior of real-time stations in legacy bus-based industrial ethernet networks”, *Computer Standards & Interfaces*, vol. 33, no. 3, pp. 249–261, Mar. 2011, ISSN: 0920-5489. DOI: `10.1016/j.csi.2010.05.002`.

[16] P. A. Laplante and S. J. Ovaska, *Real-time systems design and analysis : tools for the practitioner.* John Wiley & Sons Inc., Hoboken, New Jersey, 2012.

[17] J. Stankovic, “Misconceptions about real-time computing: A serious problem for next-generation systems”, *Computer*, vol. 21, no. 10, pp. 10–19, 1988, ISSN: 0018-9162. DOI: `10.1109/2.7053`.

[18] S. Y. Nof, *Springer Handbook of Automation.* Springer-Verlag, 2009.

[19] P. Neumann, “Communication in industrial automation—what is going on?”, *Control Engineering Practice*, vol. 15, no. 11, pp. 1332–1347, 2007, ISSN: 0967-0661. DOI: `10.1016/j.conengprac.2006.10.004`.

[20] M. Rostan, “Industrial Ethernet Technologies: Overview”, EtherCAT Technology Group, Tech. Rep., 2011.

[21] Modbus-IDA.ORG, *Modbus messaging on tcp/ip implementation guide v1.0b*, 2006.

[22] Ethernet Powerlink Standardization Group, *Ethernet powerlink*, 2014. [Online]. Available: `www.ethernet-powerlink.org`.

[23] EtherCAT Technology Group, *Ethercat technical introduction and overview*, 2013. [Online]. Available: `http://www.ethercat.org/en/technology.html`.

[24] ——, *Ethercat - the ethernet fieldbus*, 2013. [Online]. Available: `http://www.ethercat.org/pdf/ethercat_e.pdf`.

[25] Toshiba, *Tcnet technology*, 2013. [Online]. Available: `http://www.toshiba.co.jp/sis/en/seigyo/tcnet/technology.htm`.

[26] M. Felser, *Real-time ethernet - industry prospective*, 2013. [Online]. Available: `http://www.control.aau.dk/~ppm/P7/distsys/01435742.pdf`.

[27] TTA-Group, *Ttethernet - a powerful network solution for all purposes*, 2013. [Online]. Available: `http://www.ttagroup.org/ttethernet/doc/TTEthernet_Article.pdf`.

[28] CC-Link Partner Association, *Cc-link ie field brochure*, 2013. [Online]. Available: `http://am.cc-link.org/en/index.html`.

[29] ——, *Cc-link ie field white paper (clpa-2327)*, 2013. [Online]. Available: `http://am.cc-link.org/en/index.html`.

[30] R. Pigan and M. Metter, *Automating with PROFINET: Industrial Communication Based on Industrial Ethernet.* Wiley-VCH, 2008, ISBN: 9783895782947.

[31] ODVA, *The organization that supports network technologies built on the common industrial protocol (cip) - device net, ethernet/ip, componet, and controlnet*, 2014. [Online]. Available: `http://www.odva.org`.

[32] SERCOS, *International e.v. and sercos north america, group of associations dedicated to developing, promoting and expanding the use of the sercos digital interface*, 2014. [Online]. Available: `http://www.sercos.com`.

[33] industrial ethernet book, *The world market for industrial ethernet components*, 2013. [Online]. Available: `http://www.iebmedia.com/index.php?id=8595&parentid=74&themeid=255&showdetail=true&bb=true`.

[34] A. Bratukhin and T. Sauter, "Functional analysis of manufacturing execution system distribution", *Industrial Informatics, IEEE Transactions on*, vol. 7, no. 4, pp. 740–749, 2011, ISSN: 1551-3203. DOI: `10.1109/TII.2011.2167155`.

[35] K. Schmidt and E. Schmidt, "Distributed real-time protocols for industrial control systems: Framework and examples", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 10, pp. 1856–1866, 2012, ISSN: 1045-9219. DOI: `10.1109/TPDS.2011.300`.

[36] T. Hu, P. Li, C. Zhang, and R. Liu, "Design and application of a real-time industrial ethernet protocol under linux using rtai", *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 5, pp. 429–439, 2013. DOI: `10.1080/0951192X.2012.731609`. eprint: `http://www.tandfonline.com/doi/pdf/10.1080/0951192X.2012.731609`.

[37] J. Skodzik, P. Danielis, V. Altmann, and D. Timmermann, "Hartkad: A hard real-time kademlia approach", in *11th IEEE Consumer Communications & Networking Conference (CCNC)*, 2014, pp. 566–571.

[38] ——, "Time Synchronization in the DHT-based P2P Network Kad for Real-Time Automation Scenarios", in *2nd IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services (IoT-SoS)*, 2013, pp. 1–6.

[39] J. Skodzik, V. Altmann, P. Danielis, and D. Timmermann, "A kad prototype for time synchronization in real-time automation scenarios", in *World Telecommunication Congress*, 2014.

Paper B

HaRTKad: A Hard Real-Time Kademlia Approach

Jan Skodzik, Peter Danielis, Vlado Altmann, and Dirk Timmermann

In Proceedings of the 14th Annual IEEE Consumer Communications & Networking Conference (CCNC), pp. 309-314,
`https://doi.org/10.1109/CCNC.2014.6866588`, *ISBN: 978-1-4799-2356-4,*
January 2014

HaRTKad: A Hard Real-Time Kademlia Approach

Jan Skodzik, Peter Danielis, Vlado Altmann, and Dirk Timmermann
University of Rostock, Institute of Applied Microelectronics and Computer Engineering, 18051 Rostock, Germany
Email: jan.skodzik@uni-rostock.de

Abstract

The Internet of Things is becoming more and more relevant in industrial environments. As the industry itself has different requirements like (hard) real-time behavior for many scenarios, different solutions are needed to fulfill future challenges. Common Industrial Ethernet solution often leak scalability, flexibility, and robustness. Most realizations also require special hardware to guarantee a hard real-time behavior. Therefore, an approach is presented to realize a hard real-time network for automation scenarios using Peer-to-Peer technology. Kad as an implementation variant of the structured decentralized P2P protocol Kademlia has been chosen as base for the realization. As Kad is not suitable for hard real-time applications per se, changes of the protocol are necessary. Kad is extended by a TDMA-based mechanism. Additionally, to evaluate the performance an prototype is presented, which is realized on an embedded platform with a real-time operating system. Thereby, with the presented approach and a realized prototype it is possible to investigate the performance of a Kad network with hard real-time capabilities.

1 Introduction

In the field of automation, fieldbuses are widely established. They allow a deterministic data exchange and follow the requirements for hard real-time applications. However, fieldbuses are limited in address space, scalability, resilience, and interoperability. Therefore, the industry tried to use common Ethernet technology, which has technical and economical advantages in contrast to fieldbus realizations. This led to the development of Industrial Ethernet solutions, which are able to handle hard real-time constraints like fiedbuses do. However, there is no common solution available as there are even more existing Industrial Ethernet solutions on the market than fieldbuses [1]. Industrial Ethernet solutions allow for a total vertical and horizontal integration of a automation system from the field level up to company level [2]. The number of participants is usually much greater that in fieldbuses as it is only limited to MAC or IPv4/v6 addresses. Additionally, the network is not limited to any topology in contrast to fieldbuses, which often require line or ring structures. Another economical advantage is the common Ethernet technology, which is already established and makes the components cheaper in terms of development and production costs than proprietary fieldbus solutions. However, Industrial Solutions also have disadvantages. Usually they possess a central instance, which represents a single point of failure (SPoF) and bottleneck due to the realization of the system following the master-slave or server-client approach. Other realizations require dedicated and expensive hardware to realize the hard real-time behavior. Additionally, many solutions leak flexibility and need a dedicated instance for administrative tasks. These issues will becomes more relevant in the future. As mentioned in [3], the future for the industry will be more intelligent devices, which can act more dynamically. Facilities as one main area of application will consist of more devices still requiring real-time or even hard real-time behavior. So we think, the existing solutions will not fulfill the future challenges in terms of scalability, flexibility, and robustness.

Peer-to-Peer (P2P) networks instead offer an innovative alternative to the typical Client-Server or Master-Slave concepts used in Industrial Ethernet solutions. P2P runs in the application layer and thus there is no need for special hardware. As already proposed in [3], the devices like sensors/actors and other facility components become more intelligent and offer more resources.

Therefore, an P2P-based approach using Kad is presented to realize a decentralized network, which allows for hard real-time applications. The main focus is on the high robustness of the network and the scalable administration of the network. The Kad protocol has been modified to enable an arbitrary media access, which is necessary to meet timing constraints given by real-time applications. Additionally, the results of a working hard real-time Kademlia (HaRTKad) node is presented, which is required to realize a complete consistent system. The main contributions are:

- Presentation of the modified Kad protocol.
- Performance analysis of a HaRTKad client.
- Analysis of the modified Kad protocol performance consisting of HaRTKad nodes.

The remainder of this paper is organized as follows: In Section 2, the related work is presented. In the following Section 3, the basics of Kad are shortly described and the main approach to realize the HaRTKad system is explained. Section 4 describes one HaRTKad node and the realization as a prototype and its performance evaluation. In Section 5, an optimization step is presented to significantly increase the number of HaRTKad nodes to act in parallel and thus to increase the information exchange and channel utilization without violating the hard real-time constraints. Finally, the paper concludes in Section 6.

2 Related Work

Many of Industrial Ethernet solutions, like Ethercat, Ethernet Powerlink, Profinet IO/IRT, SERCOS III, and CC-Link IE Field base on the Master Slave or Client Server approach and therefore show the disadvantages of central instances. The central instance acts as

SPoF and also as bottleneck. E.g., Ethercat uses a master to synchronize the participants and to control the information exchange. Furthermore, some realizations, like Ethercat, TCnet, TTEthernet, and Profinet IO/IRT require special hardware, which is expensive and usually proprietary, which results in further constraints for a planned network. Ethercat for example needs special hardware for the slaves to achieve hard real-time performance. However, the use of special hardware for the Ethercat slaves results in expensive and proprietary hardware. Profinet IO/IRT needs a special switch for forwarding packages in the network, as a they need to follow a predefined path [4].

In [5], a real-time industrial Ethernet protocol is developed adopting a master-slave pattern. The master is developed under Linux using Real Time Application Interface and represents a SPoF. Moreover, the slaves base on universal FPGA and ARM chips so special hardware is necessary.

[6] proposes a real-time Ethernet solution, which acts without a central instance. The concept bases on two additional proprietary layers on top of the Ethernet layer to manage the media access. The result is an TDMA-approach using time slots. There is no analyses about a high number of attendees and its behavior in large scale networks. Furthermore, as TCP/UDP and IP are not supported the total vertical and horizontal is not possible without further effort.

We also propose a TDMA-based approach but directly integrate it in the Kad protocol. Thereby, we use the unchanged stack of Ethernet and UDP/IP. There are no changes needed except for the application layer. Kad will be changed to act as an application and also as an optional middle-ware for further applications on top of it. No central instance is required to configure, control, or synchronize the Kad network, which results in a well scalable, flexible, and robust network for automation scenarios, which require hard real-time constraints.

3 Basics and design concept

The distributed hash table (DHT)-network Kad (an implementation variation of the Kademlia protocol) has been used to realize a fully decentralized structured network. Every node has an own unique ID, which is called hash value. This hash value is usually generated by a hash function, e.g., the Message Digest 4 or 5 algorithm [7]. Every node is responsible for a set of data or information. The responsibility is given by the search tolerance ST. Kad generates a hash value for data or information and determines the responsible node by determine the XOR distance D between the data/information hash value and the node hash value. If the distance is less than the ST this node is responsible for the data/information. Formula 1 shows this correlation that a node is responsible if D is smaller than ST.

$$D = Hash_{Data} \oplus Hash_{Node} < ST \tag{1}$$

So the hash value directly influences, which data or information a node has to store [8].

In Industrial Ethernet solutions, the problem of fulfilling hard real-time requirements is solved by ensuring a deterministic information exchange. This is realized by using an arbitrary media access. The control of media access is done via central instances. In a TDMA-based approach, the central instance would assign time slots to the participants.

Kad instead has no central instance, which could control the media access. Therefore, a node needs to know by itself if it can access the media. A node's hash value $Hash_{Node}$ is unique and serves as information to determine time slots, during which a node is allowed to access the media. Like the relationship between data/information and the hash value, we suggest to establish a correlation between access time and the hash value as both are unique.

For high effectiveness and scalability reasons, the presented dynamic search tolerance (DST) approach from [9] is used to determine the network size and to adjust the search tolerance ST so that for hash value at least one node in the network is responsible. Contrary, this algorithm could be easily modified to guarantee that a maximum of one node is responsible for a any hash value. This new algorithm will be called inverse dynamic search tolerance (IDST). Figure 1 represents a DHT ring where the search tolerance ST_{IDST} has been determined with the IDST in such a way that maximum one node is responsible for any hash value. The new value for ST_{ISDT} will be stored consistently by the IDST algorithm on every node in the network.

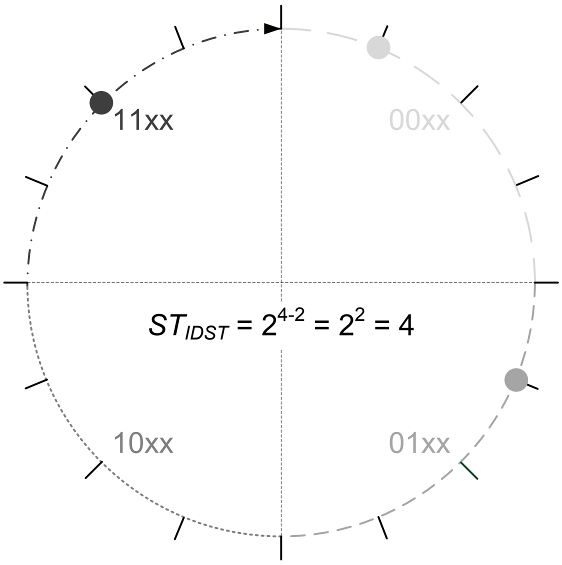


Figure 1: Result of the IDST algorithm. The whole 4 bit address space is represented by a ring. Three nodes are exemplarily placed on the ring.

A main requirement for a TDMA-based time slotted network is the common time base throughout all participating nodes. Therefore, all nodes of the Kad network need to be synchronized. An approach to synchronize a fully decentralized P2P network based on Kad without a central instance has been presented in [10]. The synchronization of a big network can be achieved very fast by using the concept of helping nodes. Additionally, the synchronization is realized as deterministic approach to make it suitable for hard real-time applications in automation scenarios.

After the IDST algorithm has been executed and the nodes are synchronized, every node knows when it is allowed to communicate (access the shared Ethernet communication medium). It is necessary to correlate the hash space of the Kad network with the time space to create the relationship between the hash values of the HaRTKad nodes and the time slots. Therefore, an approach is presented to realizes the arbitrary media access of the nodes, which is handled in a decentralized way. Every node itself knows when it is allowed to access the media without a central instance as every node has sufficient information about the time slot it belongs to. The number of slots defined in Formula 2, which are generated, directly depends on the bit width of the hash values b and the ST_{IDST}. Additionally, every network participant must run the HaRTKad application as everybody must consider its own time slot to get access to the media. Otherwise, if a network participant is not part of the HaRTKad network it could compromise the network communication and the real-time behavior cannot be guaranteed.

$$N_{Slots} = \frac{2^b}{ST_{IDST}} \tag{2}$$

The period T_{Del} represents the delivery time of a node, which can be seen as one turn around the hash ring. T_{Del} is given by Formula 3. t_{Ex} is the time needed to find an node and to exchange data.

$$T_{Del} = t_{Ex} * N_{Slots} \tag{3}$$

The actual time t_{Ring} based on the hash ring is simply expressed by Formula 4, where t_{Now} is the absolute time.

$$t_{Ring} = t_{Now} \mod T_{Del} \tag{4}$$

Now it is important for a node to know, which slot $Slot_{Node}$ it is assigned to. Therefore, Formula 5 can be used.

$$Slot_{Node} = \frac{Hash_{Node}}{ST_{IDST}} \tag{5}$$

As each node knows the value for ST_{IDST} and t_{Now}, it can decide independently if it is allowed to sent. The decision criterion is represented by Formula 6.

$$(t_{Ring} > Slot_{Node} * t_{Ex}) \wedge (t_{Ring} << Slot_{Node} * t_{Ex} + t_{Ex}) \qquad (6)$$

Using the presented formulas, we are able to realize a TDMA-based communication approach by correlating the time space with the hash space. As we are using the IDST algorithm, there is only one parameter left, which has to be determined. This parameter is t_{Ex}. t_{Ex} is a technical parameter, which is hard to determine theoretically as it depends on many parameters, especially on the used hardware. Thus, we have developed a prototype, which is presented in Section 4.

Another reason for not using the IP address of the nodes directly to assign a time slot is to support non-IP protocols on top of the Ethernet protocol as well. Therefore, we realize the assignment of the time slot in the application layer, so that we are able to support even proprietary protocol besides IP. Also, the IP address are not that equally distributed like the hash values of nodes generated by the MD4 or MD5 algorithm as they can be subnets.

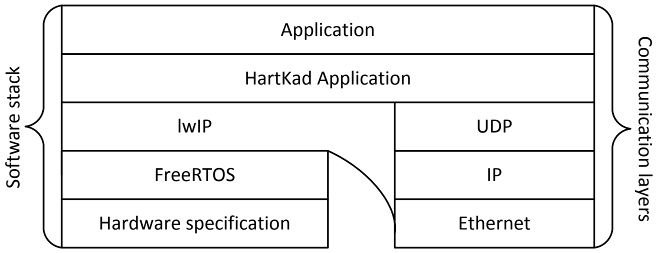


Figure 2: Software stack of a Kad node

4 Hard Real-time Kad node

As the presented approach is supposed to be used in automation scenarios, it is necessary not only to guarantee a deterministic packet exchange, i.e. communication among the nodes. It is also necessary to guarantee a hard real-time behavior of Kad node's in terms of packet processing on the nodes. Therefore, it is necessary to implement the Kad node in software using a real-time operating system. If these requirement is considered the target platform should support real-time operating systems. As target platform, the Zedboard with a 900 MHz ARM processor has been chosen [11]. We have chosen an embedded devices as we are targeting a platform for the industrial automation. Using the developed HaRTKad prototype, it is possible to determine processing times of Kad operations and transmission times of exchanged UDP packets. The software stack and communication layer of the realized prototype are depicted in Figure 2.

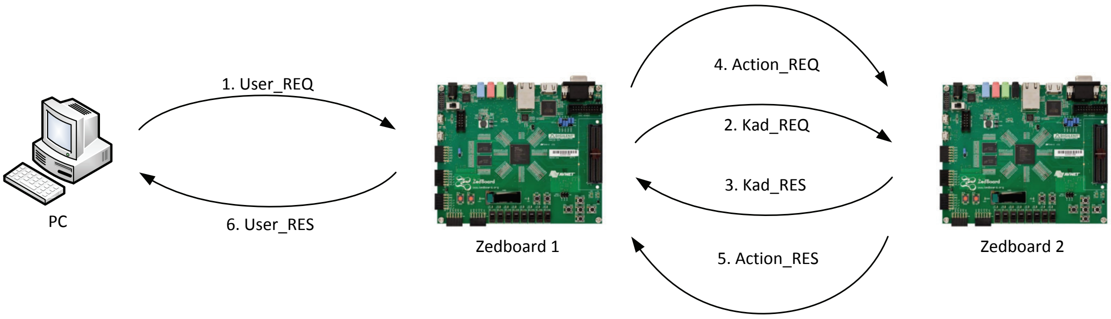


Figure 3: Prototype setup for evaluation

Software stack: First, the hardware specification of the Zedboard has to be defined in software. However, only standard hardware has been chosen for the Zedboard consisting of the ARM CPU and RAM for software. It is possible to add own dedicated hardware components, which could improve the performance. In the first approach, we renounce the use of dedicated hardware to be able to realize the system with available standard hardware. FreeRTOS has been chosen as operating system to enable hard real-time behavior of the Kad nodes [12]. This is necessary to ensure the hard real-time behavior. Additionally, lwIP is part of FreeRTOS as a lightweight TCP/IP stack to enable the communication via the Ethernet medium [13]. The next layer is the HardKad application, which enables the real-time communication and the creation of the time slots for the nodes. Here, the presented approach is realized. Additionally, it is possible to use HaRTKad as middle-ware to run further applications on top of it.

Communication layer: The communication layer consists of standard Ethernet technology as the base medium. Also, the IP protocol is supported, which is necessary to enable addressing throughout the whole infrastructure. UDP as the next layers is used as standard TCP is not suitable for real-time applications. On top, we have the application layer, which consists of the HaRTKad application and an optional additional layer for another application where HaRTKad serves as the middle-ware.

The developed nodes support multi-threaded applications, which are needed by the Kad implementation. All Kad node threads are given in Table 1 with a short description. The threads are sorted by their priority starting with the highest one.

The main thread is important to start the other threads and therefore got the highest priority. After having started the other threads, the main thread will stay idle. The external control thread got the second highest priority as it should react fast to external triggers like a human released fire alarm. The Kad communication thread has the next lower priority and is responsible for the processing of Kad packets. After the Kad communication thread, the three search threads handle the search in the Kad network. The network thread processes the packets from the network interface and forwards the

Thread	Prior.	Description
Main	5	Starts other thread before going idle
External control	4	Receiving external commands
Kad communication	3	Processing of Kad packets
Search	3	Destroys search objects
Network	2	Network interface packet processing
Maintenance	1	Maintenance threads
Idle	OS	Origin of new threads

Table 1: Threads of the Kad Client

packet data to the HaRTKad application. Three maintenance threads are responsible for keeping the network up to date. Finally, there is the idle thread, which is the source for new threads and it possesses a priority depending on the OS.

4.1 Performance Evaluation

The composite of the modified Kademlia protocol and the real-time Kad node allows to realize hard real-time applications based on P2P technology. It is necessary to build up prototype scenario of HaRTKad nodes to evaluate their performance. Therefore, a prototype setting consisting of four main components has been realized to measure the performance values. One main computer, two Zedboards and one 8-Port 1 GBit/s switch from Netgear [14] represent the prototype setup. The two Zedboards represent the Kad network and are logically connected via the Kad protocol. The setting is depicted in Figure 3. Two operations are supported in the setup, read and write operations between the two Zedboards, which are running the HaRTKad application.

Read operation: If the user requests a read operation a number of integers will be requested. The number is given by the user and stored in the user request packet. Additionally, the hash ID of the nodes, which have to deliver the requested integer values, is given. The first Zedboard receives and processes this packet. As the first Zedboard is not responsible for the read request, it starts the search in the Kad network for a node, which his responsible for the given hash ID in the read request packet. Therefore, the first Zedboard contacts the second board via a Kad request packet to see if it is still available. The second Zedboard responses via a Kad response packet. When the first Zedboard receives the Kad response, it can contact the second Zedboard again as it is also responsible for the given hash ID in the read request packet by the user. This is done via the action request packets, which is the read action request in this case. The second Zedboard answers via the Action response packet, which is the read action response packet. The read action response packet includes all integer values requested by the user and is filled with randomly created integer values. After the first Zedboard receives the read action response packet, it forwards the integer values to the user via the user response packet.

Write operation: If a write operation is performed a given number of integer values is stored in the user request packet. Like during the previous described read operation, the corresponding node for the write operation will be searched in the Kad network. In this case, it is the second Zedboard as well. This receives an action request packet, which is a write action request packet from the first Zedboard. This packet includes the integer values, which have to be stored on the responsible second Zedboard. The second Zedboard will send back a write action response packet as confirmation, which is forwarded by the first Zedboard to the user in form of the user response packet.

The times have been measured for all operations including the Kad operations and the processing of the packets. The first time measurement starts at the moment when a user request in form of a user request packet is received at the first Zedboard. The second time stamp is taken when the user request packet is sent back to the user PC. This has been measured for read and write processes. The result is depicted in Figure 4. As apparent, the results are below one microsecond. Additionally, it can be seen that with an linearly increasing number of integer values the time also increases linearly. These results can be considered for further inspection.

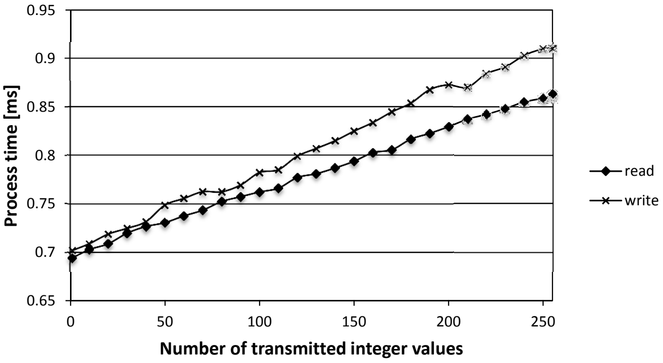


Figure 4: Performance evaluation of a HaRTKad node

5 Optimization of the channel utilization

If using a synchronous TDMA-based approach time, slots could be unsused. Therefore, one approach is presented to increase the utilization.

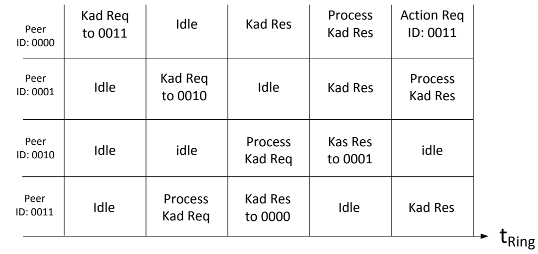
Exploiting the processing time of a node: If the best case of the processing time of a node is known this time should be considered to achieve a higher utilization of the channel. If we consider HaRTKad running on a Zedboard as target platform, it is possible to determine the channel utilization CU during the information exchange by Formula 7.

$$CU = \frac{Data_{packets}}{t_{Ex} * 1GBits/s} \tag{7}$$

It directly depends on the the data size, which has been exchanged. For the traced packets, we have six packets as previously described in the prototype scenario. The packets are summarized in Table 2 with its sizes, so that it is possible to compute the channel utilization by computing the amount of data exchanged $Data_{packets}$. For further results, only the packets inside the Kad network are considered, the packets from and to the user PC are omitted. Additionally, to make the results comparable to one scenario, we assume that only one integer value (4 Bytes) is transmitted during the a read or write process.

Table 2: Size of packets

PACKET	SUB-TYPE	SIZE [BYTE]
User_REQ		64
User_RES		64
Kad_REQ		89
Kad_RES		96
Action_REQ	read	88
	write	88
Action_RES	read	72
	write	72



Peer ID: 0000	Kad Req to 0011	Idle	Kad Res	Process Kad Res	Action Req ID: 0011
Peer ID: 0001	Idle	Kad Req to 0010	Idle	Kad Res	Process Kad Res
Peer ID: 0010	Idle	idle	Process Kad Req	Kas Res to 0001	idle
Peer ID: 0011	Idle	Process Kad Req	Kad Res to 0000	Idle	Kad Res

Figure 5: Interleaving of media access to increase the utilization

Most of the time, the nodes are processing packets and do not access the media. Thus, it is possible to allow parallel access to the media to increase the utilization. This interleaving is exemplary depicted in Figure 5. The peer with the hash ID "0000" contacts the peer with the hash ID "0011". Due to the time for processing the packets, the peer with the hash ID "0001" is able to contact the peer with the hash ID "0010".

With the determined parameters, it is possible to indicate the number of nodes, which could theoretically access the media in one time slot while considering a theoretical channel utilization of 100 %.

In a nutshell, to indicate the performance of the presented network the work of Mark Felser in [15] is taken. Three classes are defined for human control, process control, and motion control. Furthermore, these three classes have different constraints in terms of timings. For a better comparability, it is assumed that an information exchange of 4 bytes between the nodes happen. The results from our prototype shows that the the finding of a node and the exchange of 4 bytes took about 700 μs, which already includes one search step of 150 μs. Every further search step took also additional 150 μs, which is denoted with the time T_{Step}. Therefore, 550 μs are needed to send and process the action request and response packet. The number of search steps in the Kad network scales logarithmic with the total number of HaRTKad nodes as previously mentioned. It is also considered that we need more time to find other nodes in the Kad network, which is expressed by Formula 8. This is because if the optimum supports more node to achieve a higher channel utilization, we need more time to search for nodes.

$$Nodes = \frac{T_{Del}}{550us + (log_2(Nodes) * T_{Step})} \tag{8}$$

The solution of 8 is given by Formula 9, where W is the Lambert W-function.

$$Nodes_{Max} = \left\lfloor \frac{T_{Del} * log(2)}{T_{Step} * W(\frac{4}{75} * 2^{2/3} * T_{Del} * log(2))} \right\rfloor \tag{9}$$

In Table 3, the number of possible nodes is presented using the presented approach, only realized in the application layer and without any central instance.

If no optimization is realized we are only able to let a small number of nodes operate as prototypes. The amount of data, which includes the Kad packets and the action packets, is given to compute the channel utilization. Additionally, it can be seen that the actual Ethernet media utilization is very low. If we assume a theoretical channel utilization of 100% it is possible to increase the number of nodes significantly.

ATTRIBUTE	HUMAN	PROCESS	MOTION
Delivery constraint for T_{Del} [ms]	100	10	1
No optimization			
$Nodes_{Max}$	68	9	1
t_{Ex} [us]	1600	1150	700
Data amount per T_{Del} [KByte]	84.34	6.28	0.34
Channel utilization [%]	0.69	0.60	0.38
With optimization			
$Nodes_{Max}$	5110	652	90
t_{Ex} [us]	2650	2200	1750
Data amount per T_{Del} [KByte]	12799.95	1279.80	127.88
Channel utilization [%]	99.99	99.98	99.90

Table 3: Summary of the HaRTKad system performance

If we chose a human control scenario the channel utilization is only 0.69 % if no optimization is applied. Due to the low channel utilization, it is possible to increase the parallel communication. In a human control scenario, the number of nodes can be increased by the factor 75 from 68 to 5110. It is not possible to just increase the number by increasing the channel utilization von 0.69 % to 100 %, which results in a factor of 144 because with the increasing number of nodes the number of search steps also increases, which results in a higher data amount per node to find another node in the Kad network.

To realize the optimization step it is only necessary to modify the IDST algorithm, which is described in Section 3. The ISDT algorithm computes the ST_{ISDT} so that only one node is allowed to be active during a time slot. The number of slots N_{Slots} is kept and the number of nodes $Nodes_{Max}$ for the optimization is considered. The new approximated value for the nodes per slot $Nodes_{Slot\_Opt}$ for the optimization is given in Formula 10, which is close to the theoretical maximum as the hash values are well uniformly distributed on the hash ring.

$$Nodes_{Per\_Slot} = \frac{Nodes_{Max_{Opt}}}{Nodes_{Max_{No\_Opt}}} \tag{10}$$

The new $ST_{ISDT_{Opt}}$ is generated by the IDST algorithm by using the parameter $Nodes_{Per\_Slot}$.

6 Conclusion and Future Work

In this paper, an approach is presented to realize a fully decentralized Kad network enabling hard real-time operations. Additionally, the prototype for a hard real-time Kad node called HaRTKad node is presented.The combination of the presented modified Kad protocol and a working prototype allows for realization of hard real-time applications with a high resilience, flexibility, and without any SPoF. Furthermore, administrative scalability and usability in adding and removing further instances will be simplified as no managing central instance is necessary. Every node is able to act autonomous on it's given data in a deterministic way to enable the hard real-time behavior. In future, the presented approach will be used to realize an application to measure the performance in an practical scenario with a high number of HaRTKad nodes.

7 References

[1] M. Felser, "Real Time Ethernet: Standardization and implementations", in *Industrial Electronics (ISIE), 2010 IEEE International Symposium on*, 2010, pp. 3766–3771. DOI: `10.1109/ISIE.2010.5637988`.

[2] T. Sauter, "Integration aspects in automation - a technology survey", in *10th IEEE Conference on Emerging Technologies and Factory Automation, 2005. ETFA 2005.*, vol. 2, 2005, pp. 255–263. DOI: `10.1109/ETFA.2005.1612688`.

[3] P. C. Evans and M. Annunziata, "Industrial Internet: Pushing the Boundaries of Minds and Machines", General Electric, Tech. Rep., 2012.

[4] F. Klasen, V. Oestreich, and M. Volz, *Industrial Communication with Fieldbus and Ethernet.* VDE Verlag GmbH, 2011.

[5] T. Hu, P. Li, C. Zhang, and R. Liu, "Design and application of a real-time industrial Ethernet protocol under Linux using RTAI", *International Journal of Computer Integrated Manufacturing*, vol. 26, no. 5, pp. 429–439, 2013. DOI: `10.1080/0951192X.2012.731609`. eprint: `http://www.tandfonline.com/doi/pdf/10.1080/0951192X.2012.731609`. [Online]. Available: `http://www.tandfonline.com/doi/abs/10.1080/0951192X.2012.731609`.

[6] K. Schmidt and E. Schmidt, "Distributed Real-Time Protocols for Industrial Control Systems: Framework and Examples", *Parallel and Distributed Systems, IEEE Transactions on*, vol. 23, no. 10, pp. 1856–1866, 2012, ISSN: 1045-9219. DOI: `10.1109/TPDS.2011.300`.

[7] R. Brunner and E Biersack, "A performance evaluation of the Kad-protocol", *Institut Eurecom, France*, 2006.

[8] R. Steinmetz and K. Wehrle, Eds., *Peer-to-Peer Systems and Applications*, vol. 3485, Lecture Notes in Computer Science, Springer, 2005, ISBN: 3-540-29192-X.

[9] P. Danielis, J. Skodzik, V. Altmann, and D. Timmermann, "Dynamic Search Tolerance for Lookup Determinism in the DHT-based P2P Network Kad at Runtime", in *The 11th International Workshop on Algorithms, Models and Tools for Parallel Computing on Heterogeneous Platforms (HeteroPar'2013)(subm.)*, 2013.

[10] J. Skodzik, P. Danielis, V. Altmann, and D. Timmermann, "Time Synchronization in the DHT-based P2P Network Kad for Real-Time Automation Scenarios", in *The Second IEEE WoWMoM Workshop on the Internet of Things: Smart Objects and Services, IoT-SoS 2013*, 2013.

[11] Avnet. [Online]. Available: `http://www.zedboard.org/`.

[12] Real Time Engineers Ltd. [Online]. Available: `http://www.freertos.org/`.

[13] Free Software Foundation, Inc. [Online]. Available: `http://savannah.nongnu.org/projects/lwip/`.

[14] Netgear. [Online]. Available: `http://www.netgear.com/service-provider/products/switches/unmanaged-desktop-switches/GS108.aspx`.

[15] M. Felser, "Real-Time Ethernet - Industry Prospective", *Proceedings of the IEEE*, vol. 93, no. 6, pp. 1118–1129, 2005, ISSN: 0018-9219. DOI: `10.1109/JPROC.2005.849720`.

Paper C

ILP-Based Joint Routing and Scheduling for Time-Triggered Networks

Eike B. Schweissguth, Peter Danielis, Dirk Timmermann, Helge Parzyjegla, and Gero Mühl

In Proceedings of the 25th International Conference on Real-Time Networks and Systems (RTNS), pp. 8-17, `https://doi.org/10.1145/3139258.3139289`*, ISBN: 978-1-4503-5286-4, October 2017*

ILP-Based Joint Routing and Scheduling for Time-Triggered Networks

Eike Schweissguth*, Peter Danielis*, Dirk Timmermann* and Gero Mühl†
University of Rostock
* Institute of Applied Microelectronics and Computer Engineering
† Institute of Computer Science
18051 Rostock, Germany
Email: eike.schweissguth@uni-rostock.de

Abstract

Networks in the automotive and aerospace area as well as in production facilities have to support time-critical (i.e., hard real-time) communication. For such applications, time-triggered Ethernet-based networking solutions provide the required timeliness, i.e., reliable packet delivery with deterministic latencies and low jitter. However, the routing and scheduling of the time-triggered traffic is an NP-hard problem. Hence, existing solutions to this problem make certain abstractions to reduce complexity if necessary. Nonetheless, such abstractions exclude feasible routing and scheduling options from the design space. Specifically, it is a typical approach to assume a fixed routing and model the scheduling problem, only. Therefore, we present a novel ILP formulation that can be used to jointly solve the routing and scheduling problem for time-triggered Ethernet networks. Using this formulation, it is possible to solve various scheduling problems that are infeasible when using a fixed shortest path routing with separate scheduling. Compared to a fixed load balanced routing with separate scheduling, schedules computed with our formulation can offer lower communication latencies.

1 Introduction

In several application areas, the timely delivery of messages between sensors, computation units, and actors is crucial to provide reliablity and safety of the system. In the automotive and aerospace area, hard real-time communication is established by the use of bus systems like CANbus and the newer FlexRay bus or by Ethernet-based solutions like TTEthernet. In production facilities, the time-critical communication for process control and motion control tasks is provided by traditional fieldbusses or by Industrial Ethernet (IE) solutions like Ethernet Powerlink, EtherCAT, Profinet, CC-Link IE Field, TCnet, Ethernet/IP, or the aforementioned TTEthernet. A typical characteristic of these systems is the utilization of time slots dedicated to specific messages to ensure message delivery with deterministic latencies and low jitter.

In some of these systems, a time slot reflects a reservation of the whole network for a specific device (e.g., because broadcast transmissions are used). Thus, only this device is allowed to send data while all other devices in the network receive data. In this case, scheduling is rather straightforward as only a single network resource has to be scheduled. On the other hand, the use of a shared medium limits scalability and flexibility of the network because every additional network device that has to be integrated in the schedule decreases available bandwidth per device. From the networking perspective, this is inefficient resource utilization, as a broadcast transmission blocks resources for transmitting data to devices that do not need the respective data.

To overcome this problem and offer scalable and flexible networks as well as a resource management efficient enough to support higher bandwidth applications (IT and office apps), IE systems like Profinet and TTEthernet make use of switches and a traffic scheduling per link. The current standardization process of Time Sensitive Networking (TSN, IEEE802.1Q) and the research project presented in [1] also follow this communication principle. The difficulty of these systems is the high complexity of the scheduling problem, which consists of the scheduling of several messages on a large number of resources (every network link is a resource to be scheduled). The scheduling mechanism has to consider several constraints, like resource constraints of the network, or message transmission deadlines specified by the application. The final goal of the scheduling mechanism is efficient resource utilization and compliance to all applications requirements. Such a mechanism is typically realized as an offline scheduling step because of the high computational effort. Additionally, offline scheduling allows for an a priori verification of the created schedule to ensure its feasibility as well as the compliance with the given application requirements. As of the NP-hard nature of the scheduling problem, there are currently no algorithms available which solve the problem efficiently for large instances and for varying topologies, traffic patterns, and application requirements. Instead, a typical approach to the scheduling problem is the use of a formal mathematical description for all of the resource and application constraints, e.g., in an Integer Linear Programming (ILP) formulation, which can be used by different solvers to compute a feasible schedule.

Although there are several existing formulations to the given scheduling problem, e.g., as ILP, as Mixed Integer Program (MIP) or in Satisfiability Modulo Theories (SMT), it is common to make certain abstractions that reduce the computational complexity of the problem. For example, typical abstractions are the restriction of different applications to a uniform cycle time (e.g., in [2]) or the usage of time slots with predefined position and length (e.g., in [3]). While this is a comprehensible approach to create solutions that are sufficiently scalable for the intended use case, the significant reduction of the design space either reduces overall network efficiency or excludes actually feasible schedules from the design space, or both. Especially in high-load scenarios, a larger design space can offer additional scheduling options that make a problem either feasible at all or feasible with a better schedule quality.

Another possibility to offer such additional scheduling options is to jointly consider routing and scheduling. Opposed to Joint Routing and Scheduling (JRaS), existing work on time-triggered network scheduling typically assumes routes as given and/or uses network topologies that do not offer multiple routing options. Therefore, we make the following contributions within this work:

- extension of the ILP-based scheduling to a JRaS formulation
- analysis of a JRaS formulation vs. fixed routing and ILP-based scheduling using two performance indicators for solving methods: schedulability and communication latency

To the best of our knowledge, there is no work on time-triggered network scheduling that integrates JRaS in an ILP and provides a comparison of solving methods with respect to the mentioned performance indicators.

The rest of the paper is structured as follows. Section 2 summarizes related routing and scheduling approaches based on mathematical formulations. Section 3 gives a summary of the routing and scheduling problem before our JRaS ILP is introduced in Section 4. Section 5 describes our implementation and the test cases used for analysis. Our results are presented and discussed in Section 6 and Section 7, respectively. A conclusion is given in Section 8.

2 Related Work

As previously mentioned, several mathematical formulations for the scheduling problem of time-triggered networks already exist. Most notably, [2, 3] and [4] provide formal descriptions of the scheduling problem that include several similar formulas like the scheduling part of our JRaS ILP.

In [2], Hanzalek et al. focus on the scheduling for Profinet and provide a Resource Constrained Project Scheduling with Time Constraints (RCPS/TC) formulation of the problem. The authors provide a comparison between different solving techniques with respect to solving times and the quality of the determined schedule.

The work of Steiner [3] presents an SMT formulation of the scheduling problem and describes its usage in a basic solving technique that considers the complete set of scheduling constraints at once as well as its usage in an incremental solving technique with backtracking that adds and solves subsets of the scheduling constraints stepwise and thereby provides significantly lower computation times in several cases. The analysis of the solution comprises computation times and schedule efficiency in terms of resource utilization.

The ILP-based scheduling formulation given by Luka-siewycz et al. in [4] is motivated by applications from the automotive area and therefore assumes the use of the FlexRay bus. That means, there are no dedicated links that have to be scheduled. Instead, the complexity of the scheduling problem comes from a more comprehensive integration of the applications: Beyond the bus scheduling, the scheduling of computational tasks on Electronic Compute Units (ECUs) is considered.

The mentioned works differ with respect to abstractions made and the used test cases and optimization objectives. While [3] does not mention an optimization objective at all, [2] uses an objective that is mainly useful for Profinet (minimization of the red interval length) and [4] uses the end-to-end latencies of dependent tasks and messages as intuitive objective. Opposed to the formulation of Hanzalek [2], the formulations of Steiner [3] and Lukasiewycz [4] support varying cycle times for the different data flows. Nevertheless, only Lukasiewycz makes use of this feature in the used test cases. Additionally, it should be noted that Steiner and Lukasiewycz use traffic patterns in their generic test cases that include arbitrary device-to-device (D2D) communication, while Hanzalek uses simplified patterns with a single controller communicating with all other devices and D2D communication is only used in small guiding samples.

After all, the analyses of these works have a different focus than our work and mostly answer the question if the computational complexity (i.e., solver runtime) is acceptable for the intended usage area. Especially, the question of schedulability in heavily loaded scenarios is not discussed. Hence, further design options like joint routing and scheduling, which can be used to solve schedule infeasibilities, are not formulated and analyzed.

Similar routing constraints like the ones used in our JRaS ILP can also be found in the literature, as ILP- and MIP-based routing has been addressed several times, but without combining it with schedule computation for time-triggered networks. For example, [5] describes routing and wavelength assignment for optical networks. In [6], the authors discuss joint routing and scheduling for wireless mesh networks, but without using time slots dedicated to specific messages, as hard real-time constraints are not considered. The work [7] addresses hard real-time communication over Ethernet, but by the use of joint routing and bandwidth allocation.

In [8], a combined routing and scheduling approach based on Answer Set Programming modulo Theories (ASPmT) is proposed. Similar to [4], the problem formulation stems from the electronic system level and includes scheduling of computational tasks. In contrast to the work at hand, the approach only searches for one feasible solution and does not find optimal schedules for a given synthesis instance.

3 Problem Summary

To give an overview of the routing and scheduling problem, we consider a simple network topology depicted in Fig. 1. Applications running on the hosts communicate via *data flows*. The first part of the problem is to find a route from the source to the destination of the data flow. Secondly, every data flow has specific characteristics and real-time requirements, which depend on the application. Usually, control applications are considered, which use a cyclic communication and can be characterized by the following parameters:

- cycle period

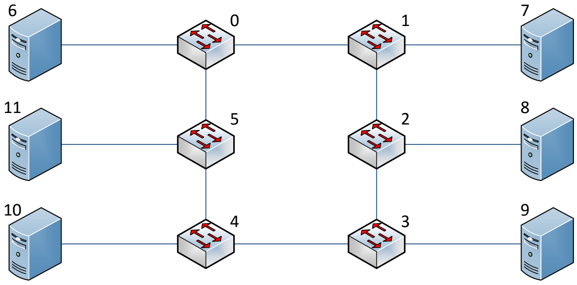


Figure 1: Sample network topology consisting of six hosts and six switches.

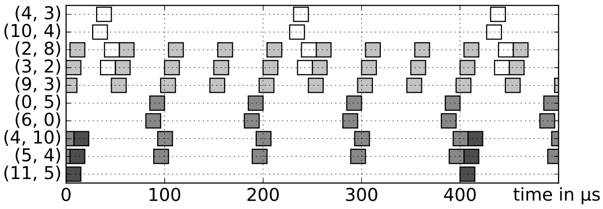


Figure 2: Schedule for four flows communicating via the network from Fig. 1. The schedule was created by our JRaS implementation and plotted using matplotlib [9].

- frame length (dependent on data amount per cycle)
- end-to-end latency requirements

While cycle period and frame length are predefined parameters, the end-to-end latency requirements must be met by the communication schedule, as higher latencies may lead to a violation of the stability criterion or performance requirements of a control task. The scheduling problem for switched, time-triggered networks considers every single network link as a separate resource for which the frame transmissions of the data flows need to be scheduled. During scheduling, resource constraints must be taken into account. Consequently, time slots on a single link must be non-overlapping and the time slots for frame forwarding on consecutive links of a route need to consider the switching delay that is imposed by the switch that connects the two links. Thus, a method solving the stated problem should be able to create schedules such as the one depicted in Fig. 2, which comprises four flows routed and scheduled in the topology from Fig. 1.

Issues of Current Solutions: In existing solutions of the stated problem, it is assumed that the routing of the flows is given, such that the mathematical problem formulation consists of pure scheduling constraints. While this approach is sufficient for network topologies that offer only a single route for every flow, it is not flexible enough for more complex topologies.

As the scheduling options depend on the given routing, the selected routing strategy can have a significant impact on the scheduling results. While a shortest path routing often offers low latencies, it can easily lead to infeasible scheduling problems if link utilization is not taken into account during routing. On the other hand, a load balanced routing may lead to higher latencies than necessary if another routing with shorter paths would also offer a feasible schedule. Besides the length of the paths, communication latencies also depend on the buffering times in the switches (i.e., the amount of time a frame spends waiting for its time slot). Therefore, it is possible that two different routings differ notably with respect to latencies, although this difference is not caused by the respective path lengths. For instance, a longer path can have a lower latency due to lower buffering times. Consequently, a joint routing and scheduling formulation is needed to take the impact of the routing into account and efficiently find the best scheduling option.

4 Joint Routing and Scheduling

In this section, we present our JRaS ILP formulation which addresses the previously mentioned problems of fixed routing approaches.

4.1 ILP Formulation

Assume that a traffic pattern is given as a set F, where every $k \in F$ is a data flow that needs to be routed and scheduled from its source s_k to its destination d_k. The network topology is given as a graph (V, E), where V is the set of nodes, containing switches and connected end devices (hosts) that are using the network. $E \subseteq V \times V$ is a set of tuples (i, j) that reflects all links that exist in the topology. The formulas of our JRaS ILP formulation can be separated into multiple subparts. Note that the major part of the ILP constraints are defined on a flow-by-flow basis, so that each flow has its own set of constraints.

4.1.1 Routing

To describe the routing, we introduce the binary variable x_{ijk}, which describes if flow k is routed via the link (i, j).

$$x_{ijk} = \begin{cases} 1\text{: flow } k \text{ is routed via link } (i,j) \\ 0\text{: flow } k \text{ does not use link } (i,j) \end{cases}$$

To start the route of a flow at its source node, exactly one more outgoing link of s_k has to be active than incoming links of s_k are active.

$\forall k \in F:$

$$\sum_{j \in V | (s_k, j) \in E} x_{s_k j k} - \sum_{j \in V | (j, s_k) \in E} x_{j s_k k} = 1 \tag{1}$$

Therefore, (1) describes the data generation of flow k at its source node s_k.

$\forall k \in F, \forall i \in V \backslash \{s_k, d_k\}:$

$$\sum_{j \in V | (i,j) \in E} x_{ijk} - \sum_{j \in V | (j,i) \in E} x_{jik} = 0 \tag{2}$$

Equation (2) reflects the forwarding behavior of all possible intermediate nodes (therefore, the constraint is not applied to source and destination of the flow). Intermediate nodes need exactly as many active outgoing links for flow k as they have active incoming links for flow k. For example, no incoming and no outgoing links are active at node i if the flow k is not routed via i. If flow k is routed via i, one incoming and one outgoing link could be active. A formula to define d_k as sink of the flow (equivalent to the source equation (1)) is not required, as d_k is the only node that is not constrained to forward the frame by (2) and thereby is implicitly defined as sink (cf. [5]). Additionally, we define (3), which, together with (1) and (2), prevents all nodes from being included in the route twice. Thereby, routing via loops is not possible.

$\forall k \in F, \forall i \in V :$

$$\sum_{j \in V | (i,j) \in E} x_{ijk} \leq 1 \tag{3}$$

4.1.2 Scheduling Along a Path

For the formulation of scheduling constraints, we introduce the following constants:

- an arbitrary but sufficiently large M
- ct_k: the cycle time of the flow k
- rsl_k: the length of a time slot required by flow k
- MSD_i: the worst case forwarding delay of a node i

Furthermore, our formulation requires the additional scheduling variables s_{ijk} and o_{ijk}.

- s_{ijk} is an integer variable with the domain $[0, ct_k - 1]$ and defines the beginning of the time slot of flow k on link (i, j) (flow **s**tart within the cycle)
- o_{ijk} is an integer variable that states in which cycle the flow k makes use of the time slot defined by s_{ijk} for the first time (flow **o**ffset)

The offset variable is necessary in several situations. For example, if a flow is scheduled at the end of a cycle (s_{ijk} close to ct_k), the delay imposed by the route to the destination may cause the time slots on the links close to the destination to fall into the next cycle. This scheduling option cannot simply be represented by a s_{ijk} with a larger domain, as this causes issues in the formulation of the resource constraints shown later. Like the routing variables x_{ijk}, the scheduling variables s_{ijk} and o_{ijk} have to exist per flow k and for every link (i, j) in the network because at the time of ILP model creation, it is still unknown which route from s_k to d_k will be used and therefore, corresponding scheduling variables have to be provided for every possible routing option.

$\forall k \in F, \forall (i, j) \in E :$

$$s_{ijk} + o_{ijk} \leq M * x_{ijk} \tag{4}$$

Equation (4) ties the scheduling variables s_{ijk} and o_{ijk} to zero for links that are no part of the currently selected route of the flow (i.e., the corresponding $x_{ijk} = 0$). For links that belong to the route, the formula is trivially satisfied because of the big M on the right side.

$\forall k \in F, \forall i \in V \backslash \{s_k, d_k\}:$

$$\begin{aligned} &\textstyle\sum_{j \in V|(i,j) \in E} (s_{ijk} + o_{ijk} * ct_k) - \sum_{j \in V|(j,i) \in E} (s_{jik} + o_{jik} * ct_k) \\ &\qquad \geq \lceil MSD_i \rceil * \textstyle\sum_{j \in V|(i,j) \in E} x_{ijk} \end{aligned} \tag{5}$$

Equation (5) defines the scheduling constraints that have to be met along a route from s_k to d_k (intra-flow dependencies). For nodes i that are not part of the route, (5) is trivially satisfied: The right side becomes zero as all outgoing links of i are inactive ($x_{ijk} = 0$), the left side is also zero because the scheduling variables of all incoming and outgoing links of i are tied to zero due to (4). For nodes i that are part of the route, the right side is simply $\lceil MSD_i \rceil$ (the sum of x_{ijk} is limited to 1 due to (3)). On the left side, the scheduling of outgoing links of i is compared to the scheduling of the incoming links. As the scheduling variables of unused links are tied to zero by (4), only the scheduling variables s_{ijk} and o_{ijk} of the links that are part of the route of flow k can contribute in each of the sums (every node that is part of the route has exactly one active incoming and one active outgoing link). This makes (5) a typical scheduling constraint: The time slot on the outgoing link has to be scheduled later than the time slot on the incoming link by at least the worst-case switching delay of the node. Additionally, note that (5) also solves an issue with the routing constraints. With the given routing constraints, it is possible that besides the actual route from s_k to d_k, useless loops are added to the routing that are completely unconnected to the actual route. These are not schedulable in (5) and are therefore excluded from the feasible solutions.

4.1.3 Resource Constraint

A schedule is only feasible if the time slots on every link do not overlap. Thus, for every pair of time slots **A** and **B** scheduled on the same link it must hold that either **A** ends before **B** starts or the other way round. These conditions are checked in Equations (6) and (7).

$\forall (i,j) \in E, \forall (k,l) \in F \times F | l > k,$
$\forall (r,t) \in \{r \in \mathbb{N} | r \leq \frac{lcm(ct_k, ct_l)}{ct_k}\} \times \{t \in \mathbb{N} | t \leq \frac{lcm(ct_k, ct_l)}{ct_l}\}:$

$$\begin{aligned} &(s_{ijl} + t * ct_l) - (s_{ijk} + r * ct_k) \\ &\geq M * (a_{ijklrt} + x_{ijk} + x_{ijl} - 3) + \lceil rsl_k \rceil \end{aligned} \tag{6}$$

$$\begin{aligned} &(s_{ijk} + r * ct_k) - (s_{ijl} + t * ct_l) \\ &\geq M * (x_{ijk} + x_{ijl} - a_{ijklrt} - 2) + \lceil rsl_l \rceil \end{aligned} \tag{7}$$

The left sides of the formulas compare the start times of two time slots while $\lceil rsl_k \rceil$ on the right side states the length of the time slot that is scheduled first. The offsets given by o_{ijk} are not considered in these formulas as they only state cycle offsets that do not matter for resource conflicts. The M-notation gives the possibility to disable constraints dynamically by trivially satisfying them by creating a sufficiently large negative value on the right side. For example, if none or only one of the currently considered pair of flows k and l is routed via the currently considered link (i, j) (corresponding $x_{ijk} = 0$), both formulas are trivially satisfied. If both flows are routed via the link, either one of the formulas (6) and (7) must be satisfied, depending on which of the two time slots under consideration comes first, while the other formula is then trivially satisfied. Which one of the formulas has to hold can be selected by the newly introduced variable a ($a = 1$: (6) has to be true). During the comparison of the time slots of two flows k and l, their cycle times must also be taken into account. If their cycle times are different, all pairs of time slots of k and l that lay within the hyper period (the least common multiple of their cycle times) must be non-overlapping. To check the respective time slots for conflicts, these cyclic repetitions within the hyper period are represented by addition of $r * ct_k$ or $t * ct_l$ to the respective time slot beginning. The upper limit of r and t respectively must be chosen such that not only all time slots of the flows k and l within their common hyper period are compared but always one additional time slot must be considered to detect conflicts at the boundaries of the hyper period. Note that the resource constraints are the reason to restrict the domain of s_{ijk} to $[0, ct_k - 1]$ and to introduce an additional offset variable o_{ijk} for the path constraints (see Section 4.1.2). If the domain of s_{ijk} was larger, the resource constraint would sometimes not detect conflicts if two flows are compared where one flow is shifted by one or more cycles (by having a $s_{ijk} > ct_k$) and the other flow is not.

4.1.4 Application Constraints

For every flow, a latency bound ml_k may be defined to restrict the communication latency from s_k to d_k.

$\forall k \in F$:

$$\begin{aligned}\sum_{\substack{j \in V| \\ (j,d_k) \in E}} (s_{jd_kk} + o_{jd_kk} * ct_k) - \sum_{\substack{j \in V| \\ (s_k,j) \in E}} (s_{s_kjk} + o_{s_kjk} * ct_k) \\ \leq \lfloor ml_k - rsl_k \rfloor \end{aligned} \tag{8}$$

Equation (8) describes this latency restriction by limiting the difference of the time slot beginnings on the last link of a route (to d_k) and the first link of a route (from s_k) to less than the application-specific latency bound ml_k minus the required transmission time on the last link (rsl_k). Equivalent to (5), only scheduling variables of links that are part of the route can contribute to the sums on the left side of the formula due to (4).

4.1.5 Optimization Objective

Finally, we can define an optimization objective for the ILP. One reasonable objective is the minimization of flow latencies, as several real-time applications benefit from lower latencies. This objective is given in (9). It consists of the sum of all flow latencies, where a single flow latency is defined by the difference of the beginning of the time slot on the last and first link of the corresponding route.

$$\min \sum_{k \in F} \left(\sum_{j \in V | (j, d_k) \in E} (s_{jd_k k} + o_{jd_k k} * ct_k) - \sum_{j \in V | (s_k, j) \in E} (s_{s_k jk} + o_{s_k jk} * ct_k) \right) \tag{9}$$

4.2 Model Optimization by Route Preprocessing

While the formulation from Section 4.1 is fully functional, it creates several unnecessary variables and constraints: The variables x_{ijk}, s_{ijk} and o_{ijk} are created for every link/flow combination although for a flow k, there are generally several links (i, j) that can never be part of a correct routing because the presented routing constraints allow only simple (i.e., loop-free) paths. For example, outgoing links from a destination, incoming links to a flow source or links to leaf nodes that are not the destination can never be feasible routing options. Therefore, it is not necessary to create x_{ijk}, s_{ijk} and o_{ijk} for such links. Furthermore, corresponding constraints can be left out. For instance, the number of Equations (2), (3), (4), and (5) is reduced. The number of resource constraints is also reduced, because Equations (6) and (7) only need to be added to the ILP model for a link (i, j) if both flows of the pair of flows (k, l) have this link as possible routing option. According to these considerations, we optimized our ILP model creation by a route preprocessing step. In this step, all simple paths for every source-destination pair of the traffic pattern are computed so that they can be considered during ILP model creation such that only required variables and constraints are added.

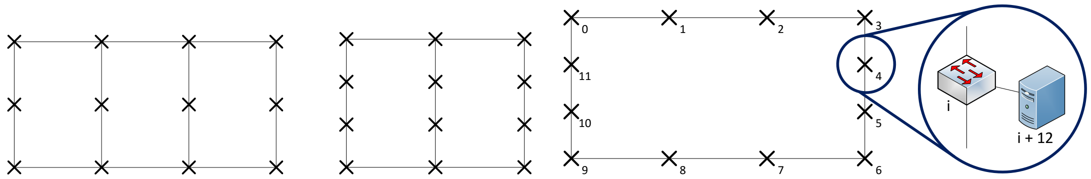
Although ILP solvers like Gurobi have a presolve step that is able to remove redundant variables and constraints from the ILP model before starting the optimization, we were able to further reduce the size of the *presolved* model by about 50% by adding the route preprocessing step. In the results, our JRaS implementation with route preprocessing is abbreviated as *jras*.

5 Test Setup

In our analysis, we use several test cases with generic traffic patterns. Details on the implementation and the test case creation are given in the following subsections.

5.1 Implementation

We implemented our JRaS ILP in a Python-based framework, in which we can create topologies as *networkx* [10] graph, generate or manually specify different traffic patterns, and use the Python API of the Gurobi Optimizer [11] (Version 7.0.1) to create and solve the respective ILP instances. All computations are executed on an Intel Xeon E5-2667v2 processor with 128 GB RAM, and with 4000 seconds configured as solver time limit for a single optimization run.



(a) PM4-3-2 (b) PM3-4-3 (c) Ring topology with node IDs of the switches.

Figure 3: Topologies used in our test cases. PM4-3-2 stands for partial mesh with 4 columns, 3 hosts per column and horizontal links every 2 rows. Node ID of a host is the node ID of the switch it is connected to plus twelve.

5.2 Topologies

The topologies used in our test cases all include 12 switches and 12 hosts and are depicted in Fig. 3. They include a ring structure, which is a typical example of an automation network topology, and two partial mesh topologies that have a slightly higher connectivity and thereby provide further routing options. We did not consider topologies with a higher connectivity (e.g., FatTree, 2D Mesh Grid) as we wanted to demonstrate the differences of the solving methods in topologies with a reasonable cabling effort if used in practical setups. As regards device characteristics, the delay per switch is assumed as 4 μs, which is a realistic assumption for state-of-the-art cut-through switches [2].

5.3 Test Cases With Generic Traffic Patterns

We define the traffic classes depicted in Table 1 and the following rules for the generation of random traffic patterns:

- every data flow k added to a pattern F is a simple one-way communication flow
- source and destination of each data flow are randomly selected but the likelihood of communication between two hosts is exponentially decreasing (factor 0.9) with their distance (in terms of hops), which leads to a preference of local communication and thereby less high-distance flows are part of a pattern
- all data flows of a pattern F are portioned over the four traffic classes as stated in the last column of Table 1

For a single test case, we generate 100 traffic patterns with the same number of flows per traffic pattern. Then, we solve the routing and scheduling problem for each pattern with different solving methods (jras and comparative approaches that are described in Section 6.1). There are four possible solver outcomes for every optimization run:

1. solved, optimal schedule found
2. solved, time limit reached, sub-optimal schedule found
3. unsolved, time limit reached without a feasible schedule

Table 1: Traffic Classes.

Class	Slot Length [μs]	Cycle Time [μs]	Max. Latency [μs]	Flowcount within a pattern F
1	15	50	200	$\lfloor 0.2 * \lvert F\rvert \rfloor$
2	15	100	400	$\lfloor 0.2 * \lvert F\rvert \rfloor$
3	15	200	800	$\lfloor 0.3 * \lvert F\rvert \rfloor$
4	15	400	1600	$\lvert F\rvert - \lfloor 0.3 * \lvert F\rvert \rfloor - 2 * \lfloor 0.2 * \lvert F\rvert \rfloor$

4. unsolved, infeasibility detected (cannot be scheduled such that all constraints are satisfied)

6 Results

In this section, we analyze to what extent multiple solving methods differ in terms of the percentage of successfully scheduled traffic patterns (solver outcome 1) and 2)) and in terms of flow latencies for the resulting schedules.

6.1 Fixed Routing and Scheduling (FRaS)

We compare our JRaS approach to two-step solutions that compute routes first and then apply ILP scheduling to the fixed, pre-computed routes. Like jras, these solving methods were implented in our Python-based framework as described in Section 5.1. An ILP model for scheduling only can easily be derived from the JRaS ILP given in Section 4.1 and basically reflects the results achievable with the scheduling formulations presented in the related work. For route selection, we make use of two strategies.

6.1.1 Shortest Path Routing

The first route selection strategy uses shortest path routing with simple load balancing. If there are multiple shortest paths for a source-destination pair and also multiple traffic flows for that pair, the traffic flows are assigned to the different shortest paths in a round-robin scheme. Additionally, if there are multiple links between two nodes (i, j), all flows that are routed from i to j are assigned to the available links in a round-robin scheme. In the results, the solution with fixed shortest path routing and separate scheduling is called *sps*.

6.1.2 Load Balanced Routing

The second fixed routing used in our tests is a load balanced routing that is optimized for a low maximum link utilization. We also use an ILP formulation for this routing approach, which uses Equations (1) - (3) as routing constraints. Additionally, the constraint given in (10) introduces the variable $MaxLinkUtil$ as upper bound of all link utilizations.

$\forall (i,j) \in E:$

$$\sum_{k \in F} x_{ijk} * \frac{rsl_k}{ct_k} \leq MaxLinkUtil \tag{10}$$

The optimization objective of the load balanced routing ILP is stated in (11). Due to the strong weighting of $MaxLinkUtil$, it basically leads to the lowest possible maximum link utilization.

$$\min \sum_{(i,j,k) \in E \times F} x_{ijk} + MaxLinkUtil * 1000 * |E| * |F| \tag{11}$$

If there are multiple routing options with the same value for $MaxLinkUtil$, the additional sum of link usages x_{ijk} leads to a preference of the routing option which uses shorter paths. The solution with this fixed routing and separate scheduling is abbreviated as *lbal*. A summary of the three solving methods used in the tests is given in Table 2.

Table 2: Solving Methods Used in the Tests.

jras	ILP-based joint routing and scheduling
lbal	fixed load balanced routing and ILP-based scheduling
sps	fixed shortest path routing and ILP-based scheduling

6.2 Load Dependency

The first four test cases show how the difference between the solving methods depends on the network load. They use the topology from Fig. 3 (a) and varying network load from 20 flows per pattern to 35 flows per pattern, with a step size of 5 flows.

At first, the results (Fig. 4) show that the test cases represent the transition from traffic patterns that are easy to schedule to traffic patterns that have a higher chance of being infeasible due to highly utilized network links. The infeasibility is either due to resource constraints, i.e., link over-utilization and corresponding violation of Equations (6) and (7), or because highly utilized links offer only time slots for a specific flow that impose long buffering times (up to one flow cycle time) at one or multiple switches, such that the maximum latency constraint (8) is violated for that flow. A clear answer to the question what causes the infeasibility would require the computation of the Irreducible Inconsistent Set (IIS) and its analysis for every single traffic pattern, which is out of the scope of this work.

In scenarios with higher network load, the probability of highly utilized links that cause one of the mentioned issues is increasing. Then, a load balanced routing becomes more important because it can offer a lower link utilization, which avoids issues with resource constraints and may also reduce the buffering times. As sps is the only solution which does not include effective load balancing, the relative advantage of jras and lbal vs. sps

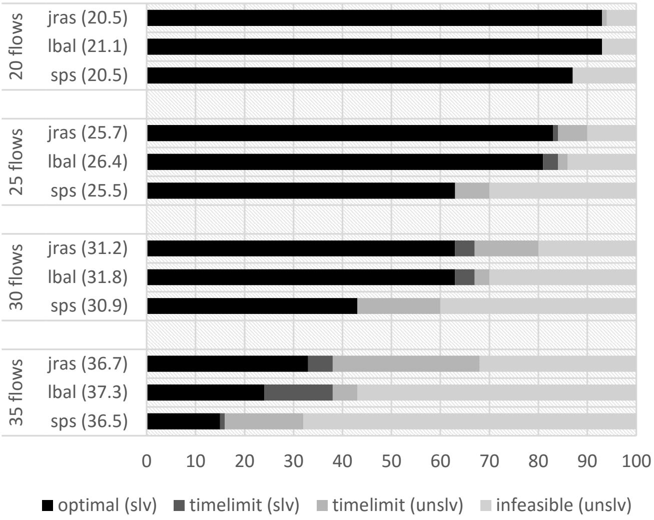


Figure 4: Results for the three solving methods for the PM4-3-2 topology and varying load. Numbers in parentheses state the total network utilization in percent (average over solved patterns).

in terms of the number of patterns with a feasible solution is increasing from 20 flows per pattern to 35 flows per pattern.

With respect to latency, the comparison of sps and jras latencies (Fig. 6) shows no benefits for jras. Thus, we can derive that, at least for the traffic patterns used, it is possible to find schedules for the shortest path routing that do not involve high buffering times. Otherwise, jras would provide schedules with lower latencies by switching to a different routing with reduced buffering times.

Furthermore, Fig. 6 shows that lbal is outperformed by the other solving methods. It has up to 3.3 % higher latencies on average and up to 61.8 % in the worst case, as the load balanced routing often includes non-shortest paths, which are sub-optimal with respect to latency. In such cases, the jras design space offers the opportunity to select a routing with shorter paths and thereby lower latencies if such a routing still provides feasible scheduling options.

6.3 Topology Dependency

Secondly, we analyze the influence of the topology on the performance of the different solving methods by comparing the results for all three topologies from Fig. 3. For every topology, we run a single test case which consists of 100 randomly generated patterns and 30 flows per pattern.

The results of the test cases are shown in Fig. 5. In this figure, we can see an increasing network load from top to bottom, although the considered test cases use the same number

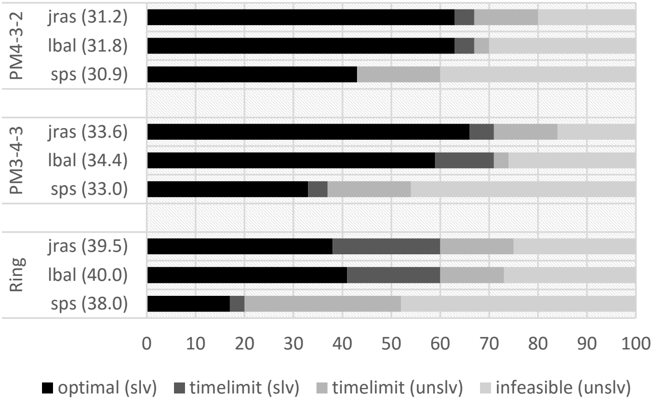


Figure 5: Results for 30 flows per pattern and varying topologies.

of flows per pattern and the same traffic classes. This is reasoned by the fact that the lower connectivity of the PM3-4-3 and Ring topology leads to less routing options and a higher average hop distance per host pair. Therefore, a single flow creates a higher network load by using more links. While our pattern generation methodology weakens this effect by making large distance communication less likely, Fig. 5 still clearly shows the influence of the topology and especially the significant drawback for the simple ring topology. Again, the general schedulability decreases with increasing network load, while the advantage of jras and lbal vs. sps becomes more prominent. Likewise, lbal is outperformed by sps and jras in terms of latency and shows up to 3.1 % higher latencies on average and up to 34.3 % higher latencies in the worst case. It is important to note that these test cases demonstrate that JRaS can be very useful even if the topology does not provide many routing options, like the ring topology.

6.4 Control Traffic

Finally, we considered a test case with a different traffic pattern generation: To reflect typical control traffic patterns, we selected two of the hosts as control nodes, which receive exactly one data flow from each other host and send one data flow to each other host. A special traffic class (Slot Length 8 μs, Cycle Time 100 μs, Max. Latency 400 μs) is assigned to all of these flows. Additionally, eight more data flows, which are generated with the previously used rules and belong to the first four traffic classes, are added to a pattern. Thereby, each of the 100 patterns of this test case consists of 52 flows. The topology used is a simple ring structure (Fig. 3 (c)) and hosts 12 and 13 were used as control nodes. Furthermore, the topology has duplicated links between the node pairs (0,1), (0,12) and (1,13) to adjust to the significantly higher network load in the area of the control nodes.

In the results of this test case (Fig. 7), the most significant observation is that jras even outperforms lbal by far in terms of schedulability. We assume that this can be explained

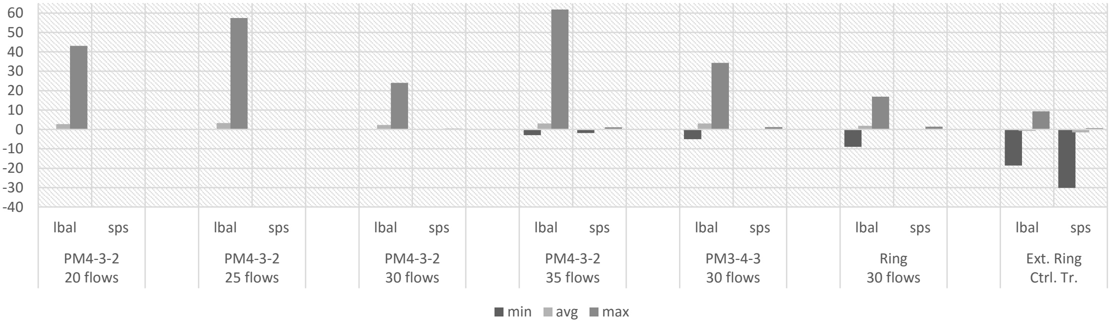


Figure 6: Latency comparison based on the objective value defined in Eq. (9). The objective value of jras is used as reference. Then, for every pattern where jras and the respective solving method under comparison both have a feasible schedule, the relative difference of the objective value is computed. Minimum, maximum, and average of the relative difference are analyzed per test case. Missing bars are intentional and represent zero difference to the jras reference.

by the fact that data flows can be assigned to a link in a way that the link utilization is below 1 (i.e., there is no over-utilization), but the selected data flows can still not be scheduled on the same link due to the composition of slot lengths and cycle times. This issue is especially present when varying slot length and cycle times are used and it is not addressed by the load balanced routing. In constrast, jras can find load balanced routings that are compatible with respect to slot lengths and cycle times and can therefore be scheduled successfully.

The latency comparison of this test case shows high differences between the solving methods in both directions (min and max values) with a slight drawback for jras on average. This is reasoned by the fact that this test case shows a high load scenario and a lot of latency values under comparison are sub-optimal solutions. These can reflect arbitrary solutions from the large design space and therefore, the latency results for this test case have a high variance. As jras is the only solving method which includes solutions with rather inefficient routing in its design space, this can lead to slightly higher latencies. Nevertheless, preferring jras as solving method in this case can easily reasoned by the significantly better schedulability compared to both other solving methods. It should also be considered that better latency results can be expected by jras if a higher time limit is used.

7 Discussion

Our analysis has shown that each of the two FRaS solving methods yields good results with respect to one of the two performance indicators. While sps can generally provide low latencies in our test cases, a comparatively low fraction of traffic patterns is schedulable by sps. The lbal approach has a high percentage of schedulable traffic patterns, but

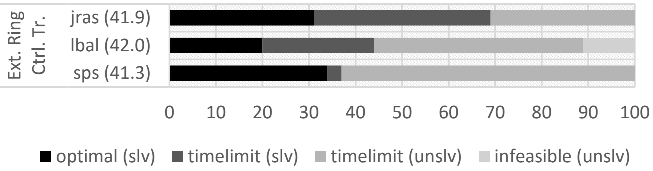


Figure 7: Results for an extended ring topology with control traffic.

Table 3: Average Solver Runtimes for Optimally Solved Patterns.

	PM4-3-2 20 flows	PM4-3-2 25 flows	PM4-3-2 30 flows	PM4-3-2 35 flows	Ring 30 flows	Ext. Ring Ctrl. Tr.
jras	11.99 s	73.97 s	213.09 s	816.67	455.20 s	1284.53 s
lbal	1.39 s	9.00 s	93.98 s	90.95	177.35 s	1114.51 s
sps	0.40 s	1.10 s	95.72 s	345.43	68.89 s	406.73 s

cannot keep up with the other solving methods with respect to latency, especially when using traffic patterns that are still schedulable with a shortest path routing due to a low network utilization.

Opposed to the FRaS solving methods, the JRaS approach from Section 4.1 performs well with respect to both performance indicators. This shows that additional scheduling options offered by a comprehensive modeling approach like JRaS cannot be neglected if optimal scheduling results shall be achieved. The joint consideration of routing and scheduling provides the flexibility to find solutions with short paths and low latencies in case of traffic patterns with low network utilization and with an appropriate load balancing in case of traffic patterns with higher network utilization. On the other hand, the significantly larger design space of JRaS, which enables these advantages, imposes a higher model complexity. Table 3 gives an overview of the complexity differences of the solving methods by showing a runtime comparison for optimally solved traffic patterns of several test cases. Still, it should be noted that these runtimes have a very large variance and therefore only represent a rough estimate of the complexity.

Although an FRaS approach offers a lower complexity, the results show that this is not worth the trade-off with respect to scheduling performance: For low network utilization, solver runtimes for jras are higher compared to lbal and sps, but generally not problematic. For high network utilization, the relative runtime advantage of the FRaS approaches decreases notably, while their scheduling results become unacceptable compared to JRaS, which makes JRaS the more reasonable scheduling approach. However, it becomes evident that larger problem instances than considered here will create runtime issues for JRaS, when solved by a state-of-the-art ILP solver.

Furthermore, the given JRaS ILP formulation can be extended by complex application constraints that were already discussed in the related work. For example, multicast flows

[3] [2], inter-flow dependencies [3] [4] and even the task scheduling on computational resources [4] for tasks that produce or consume flow data can be integrated into the model. After all, such a model with JRaS and extended application constraints will be too complex to schedule problems of reasonable size with current ILP solvers.

Finally, a need for more advanced scheduling strategies becomes apparent, such that future large scale networks with hundreds of devices and thousands of flows can be scheduled with reasonable computation times and with consideration of complex application constraints. Such new scheduling strategies should leverage more efficient design space exploration strategies to improve scalability, while a reduction of the design space by abstractions like fixed routing or time slots of fixed length and position should be applied only under careful consideration of the drawbacks with respect to scheduling performance.

We also identified the development of meaningful benchmarks for a quantitative comparison of different solving methods as a future task. The results of recent works are hardly comparable for several reasons (compare Section 2): The formulations differ in constraints (i.e., also in terms of abstractions made to reduce complexity) and optimization objectives and their selected test cases differ in terms of used topologies, traffic patterns and scale. Additionally, large scale tests are usually based on generic traffic patterns and future analyses need to verify if the results still hold for real world applications, as topologies and traffic patterns have a significant impact on the results.

8 Conclusion

We successfully integrated JRaS into an ILP formulation and verified its applicability for scheduling time-triggered Ethernet networks. In its current state, our approach is usable for TTEthernet and TSN with time-aware traffic shaping, and it can be modified for scheduling of Profinet with minor effort. We created test cases that enable a comparison of our JRaS approach to other scheduling methods that use a fixed routing. Obtained results are analyzed with respect to two important aspects: schedulability and communication latency. In these test cases, we used generic traffic patterns with medium to high overall network load and realistic but highly demanding flow characteristics. For such scenarios, the fixed routing approaches either have drawbacks with respect to schedulability or latencies, while JRaS yields good results with respect to both of these aspects in all test cases. The scheduling approach with fixed load balanced routing could keep up with JRaS in terms of schedulability, but showed up to 61.8 % higher flow latencies. Contrary, the scheduling approach with fixed shortest path routing could provide good latency results, but in the worst case, only 33.3 % of the traffic patterns that were schedulable by JRaS could be successfully scheduled. Based on our results, it becomes clear that a comprehensive modeling approach like JRaS is the method of choice for routing and scheduling of time-triggered networks. Furthermore, we derived the need for new scheduling strategies that take the benefits of the JRaS approach into account and consider complex application constraints as well, because current scheduling strategies are not sufficiently scalable to solve such a comprehensive model for large-scale networks. During the development of these new scheduling strategies, it will be necessary to refine the benchmarks used, such that different approaches can be compared in a meaningful way.

9 References

[1] E. Schweissguth, P. Danielis, C. Niemann, and D. Timmermann, "Application-aware industrial ethernet based on an sdn-supported tdma approach", in *IEEE World Conference on Factory Communication Systems (WFCS)*, 2016, pp. 1–8.

[2] Z. Hanzalek, P. Burget, and P. Sucha, "Profinet io irt message scheduling with temporal constraints", in *IEEE Transactions on Industrial Informatics*, vol. 6, 2010, pp. 369–380.

[3] W. Steiner, "An evaluation of smt-based schedule synthesis for time-triggered multi-hop networks", in *IEEE Real-Time Systems Symposium (RTSS)*, Dec. 2010.

[4] M. Lukasiewycz, R. Schneider, D. Goswami, and S. Chakraborty, "Modular scheduling of distributed heterogeneous time-triggered automotive systems", in *Asia and South Pacific Design Automation Conference (ASP-DAC)*, Feb. 2012.

[5] B. Jaumard, C. Meyer, and B. Thiongane, "Ilp formulations for the rwa problem: Symmetric systems", *published on researchgate.net*, Nov. 2004.

[6] J. Zhang, H. Wu, Q. Zhang, and B. Li, "Joint routing and scheduling in multi-radio multi-channel multi-hop wireless networks", in *IEEE International Conference on Broadband Networks*, Oct. 2005.

[7] J. W. Guck and W. Kellerer, "Achieving end-to-end real-time quality of service with software defined networking", in *IEEE Intl. Conf. on Cloud Networking (CloudNet)*, Oct. 2014, pp. 70–76.

[8] K. Neubauer, C. Haubelt, P. Wanko, and T. Schaub, "Enhancing symbolic system synthesis through aspmt with partial assignment evaluation", in *Proceedings of Design, Automation and Test in Europe (DATE) (to appear)*, Mar. 2017.

[9] J. D. Hunter, "Matplotlib: A 2d graphics environment", *Computing In Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007. DOI: `10.1109/MCSE.2007.55`.

[10] A. A. Hagberg, D. A. Schult, and P. J. Swart, "Exploring network structure, dynamics, and function using networkx", *Proceedings of the 7th Python in Science Conference (SciPy2008)*, pp. 11–15, Aug. 2008.

[11] I. Gurobi Optimization, *Gurobi optimizer reference manual*, 2016. [Online]. Available: `http://www.gurobi.com`.

Paper D

Dynamic Flow Migration for Delay Constrained Traffic in Software-Defined Networks

Peter Danielis, György Dán, James Gross and André Berger

In Proceedings of the IEEE Global Telecommunications Conference (GLOBECOM) 2017, pp. 1-7, `https://doi.org/10.1109/GLOCOM.2017.8254753`*, ISBN: 978-1-5090-5019-2, December 2017*

Dynamic Flow Migration for Delay Constrained Traffic in Software-Defined Networks

Peter Danielis*, György Dán†, James Gross† and André Berger‡
* Faculty of Computer Science and Electrical Engineering, University of Rostock, Germany
† School of Electrical Engineering and the ACCESS Linnaeus Center, KTH Royal Institute of Technology, Sweden
‡ School of Business and Economics, Maastricht University, The Netherlands
E-mail: peter.danielis@uni-rostock.de,{gyuri,james.gross}@kth.se, a.berger@maastrichtuniversity.nl

Abstract

Various industrial control applications have stringent end-to-end latency requirements in the order of a few milliseconds. Software-defined networking (SDN) is a promising solution in order to meet these stringent requirements under varying traffic patterns, as it enables the flexible management of flows across the network. Thus, SDN allows to ensure that traffic flows use congestion-free paths, reducing the delay to forwarding and processing delays at the SDN nodes. However, accommodating new flows at runtime is under such a setting challenging as it may require the migration of existing flows, without interrupting ongoing traffic. In this paper, we consider the problem of dynamic flow migration and propose a polynomial time algorithm that can find a solution if direct flow migration is feasible. We furthermore propose an algorithm for computing both direct and indirect flow migration and prove its correctness. Numerical results obtained on a FatTree network topology show that flow migration is typically necessary for networks with a moderate number of flows, while direct flow migration is feasible in around 60% of the cases.

Index terms— SDN, dynamic flow migration, routing.

1 Introduction

Future networks are facing new challenges due to new application requirements, mainly driven by the ongoing digitization drive in factory automation and process control and by emerging tactile applications, which has triggered an increasing interest in low-latency networking [1]. As an example, in industrial automation environments, a wide range of applications require a reliable real-time (RT) communication system that ensures the timely delivery of sensing or actuation data. A violation of communication deadlines may lead to unacceptable consequences like damage imposed on parts of the production facility. Recently, Industrial Ethernet (IE) systems, which can guarantee RT of below 1 ms, have emerged as serious competitor to traditionally used fieldbusses. As pointed out in [2], each IE system has however at least one drawback such as limited scalability in terms of the number of devices, insufficient self-configuration features, or increased costs due to the use of proprietary hardware instead of standard Ethernet hardware.

Therefore, previous works have developed new RT communication systems that overcome the drawbacks of the existing IE systems while providing comparable latency characteristics (below 1 ms) and independence of the used network topology [3, 4]. These works leverage the approach of Software-Defined Networking (SDN) and exploit the SDN controller's central view on the network to discover the topology and to collect the requirements of the applications using the network. Then, the collected information is used to compute paths for all required network flows that ensure compliance with the application requirements (e.g., bandwidth, maximum latency). The features provided by existing SDN technologies like OpenFlow are used to install the customized, application-specific flows in the network. However, the systems are currently solely able to compute paths for a fixed network configuration.

Extending these previous works, in this work we consider the problem of flow migration assuming unsplittable flows like in [5] using edge-disjoint paths, and adding one flow at a time. We propose novel algorithms for dynamic flow migration that migrate paths at runtime to accommodate for the new flow without interrupting ongoing flows. We use the proposed algorithms to address three fundamental questions related to the flow migration problem:

1. In what scenarios is a flow migration necessary to be able to accommodate a new flow?
2. If a flow migration is necessary, how many migration steps are necessary to accommodate a new flow and what is the proportion of direct flow migrations compared to indirect flow migrations?
3. How computationally expensive is flow migration as a function of the number of existing flows?

The rest of this paper is organized as follows. Section 2 gives an overview of SDN and OpenFlow and discusses related work. Section 3 presents the system model and the problem formulation. Section 4 describes the algorithms for direct and indirect flow migration. Numerical results are shown in Section 3, and Section 7 concludes the paper.

2 Background and Related Work

2.1 SDN for Hard Real-time Communication

The SDN approach simplifies traditional network infrastructure devices (NIDs) by moving the control logic to a central SDN controller [6]. It is the SDN controller that computes routing decisions, and installs appropriate rules on NIDs to implement routing decision.

Unlike the traditional distributed control logic, the implementation of essential parts of the control logic, such as topology discovery is thus facilitated due to the controller's overview of the network. The interface between the SDN controller and the NIDs used to transfer rules is referred to as the Southbound API. Moreover, the controller usually provides for a Northbound API, which enables the communication with applications in the network, which could be used by the controller to learn the delay constraints of all applications.

One widely used protocol for conveying information over the Southbound API is the OpenFlow protocol [7] [6]. We will subsequently refer to OpenFlow-capable NIDs as OpenFlow switches. In addition to the specification of the communication between OpenFlow switches and the SDN controller, OpenFlow further defines the composition of forwarding rules. A forwarding rule is installed on a switch and will be applied to each incoming packet that contains matching header fields. If a switch does not have a matching forwarding rule available it requests an appropriate rule from the controller. It can be assumed that future IE switches support OpenFlow since it has already become the de-facto Southbound API standard [7, 8].

Owing to the advantages of the centralized control logic, SDN has recently been proposed as a means of enabling hard-real time communication for industrial control applications [3, 4]. The proposed solutions either use SDN for assigning time slots to flows across the network [4], or for enforcing flow-specific bandwidth allocations combined with static priority queuing in the switches [3]. While these works show the feasibility of using SDN for hard real-time communication, they have in common that they do not support changing networks, thus they do not allow adding new devices and flows into the network at runtime, without potentially interrupting existing flows. Our work addresses this gap, and could be used in combination with the flow-specific bandwidth allocation mechanism proposed in [3] for accommodating new flows.

2.2 Related Work

Flow migration (FM) in SDN has recently received some attention [9, 10, 5, 11]. In [9], each packet or flow is stamped with a version number to refer to old or new forwarding rules, and a packet/flow is solely routed with regard to one set of rules. Unfortunately, this approach does not always ensure a conflict-free FM in the case of migrating multiple flows at once. The SWAN approach is presented in [10], which finds conflict-free FM if a fraction of capacity (slack) is left on all paths. However, this assumption in not realistic in practice. The authors in [5] try to find a consistent migration ordering by executing a search in a dependency graph of possible migration steps. Like our work, they also assume unsplittable flows, which are only allowed to migrate as a whole to another path. They show the corresponding decision problem to be NP-hard under switch memory constraints. Unlike the algorithm we propose, the algorithm in [5] does not always find a conflict-free FM if a temporary flow migration to a path is required to achieve the final FM. To address this, certain flows are assumed rate-limited to enable a conflict-free FM.

Brandt et al. show that given a network flow configuration, there is a polynomial-time algorithm to decide if a congestion-free migration is possible for the case of splittable flows [11]. However, they mention that the decision problem is NP-hard if all flows must be integer or are unsplittable.

Our work extends these recent works by proposing a polynomial time algorithm for deciding whether direct FM is feasible, and by evaluating the necessity and complexity of indirect FM. Although our work focuses on the algorithms for calculating forwarding rules, we also suggest a strategy how to deploy them on switches, namely changing the rules of one switch at a time. As opposed to the work in [12], which proposes to compute a network update schedule in order to minimize the overall network update time, our approach does not require switches to be synchronized.

3 System Model and Problem Formulation

We consider an SDN-enabled network and model it by a directed graph $\mathcal{G} = (V, E)$, where V is the set of vertices (the switches and the hosts), and E is the set of edges. There is a set $\mathcal{C} = \{(s_i, t_i) : 1 \leq i \leq N, s_i, t_i \in V\}$ of N source-destination host pairs that want to exchange data over the network. For a source-destination pair (s_i, t_i), $s_i, t_i \in V$, we call a sequence of edges in $\mathcal{G}$ from s_i to t_i a path, and define the length of path P, denoted by $l(P)$, to be the number of edges it contains.

We consider that the traffic from s_i to t_i has a specific delay constraint, and for simplicity we consider that the delay constraint can be met if the path from s_i to t_i has length at most $L_i \in \mathbb{N}^+$. For a pair (s_i, t_i) we define the set $\mathcal{P}_i = \{P_i | l(P) \leq L_i\}$ of length-constrained paths, and we use the notation $P_i \in \mathcal{P}_i$ to refer to a member of this set. Clearly, any of the paths $P_i \in \mathcal{P}_i$ allows s_i to send data to t_i. Given the set $\mathcal{C}$ of source-destination pairs, we say that a collection of paths $R = \{P_1, \ldots, P_N\}$ is valid if P_i and P_j are mutually edge-disjoint, and we refer to it as a route. A route allows the source-destination pairs in $\mathcal{C}$ to communicate simultaneously. Finally, we denote the set of all possible routes for the host pairs in $\mathcal{C}$ by $\mathcal{R}_\mathcal{C}$. Figure 1 illustrates the model, not showing the directions of the edges for simplicity.

We consider that the SDN controller has global information and can be used to update the forwarding rules of one switch at a time, e.g., using the SDN Southbound API. We refer to this one-step updating process in the following as *atomic forwarding table update*. The restriction of changing the rules of one switch at a time avoids unpredictable network behavior due to race conditions caused by imperfect synchronization between the switches. Atomic forwarding table updates could thus serve as atomic steps for lightweight path- and route migration, defined as follows.

Definition 1. *A flow migration (FM) from route R to route R' is a sequence $\mathcal{S} = (R^{(0)}, \ldots, R^{(K)})$ of $K \geq 1$ routes $R^{(k)}$, such that $R^{(0)} = R$ and $R^{(K)} = R'$, and route $R^{(k)}$ can be obtained from route $R^{(k-1)}$ by a sequence of atomic forwarding table updates.*

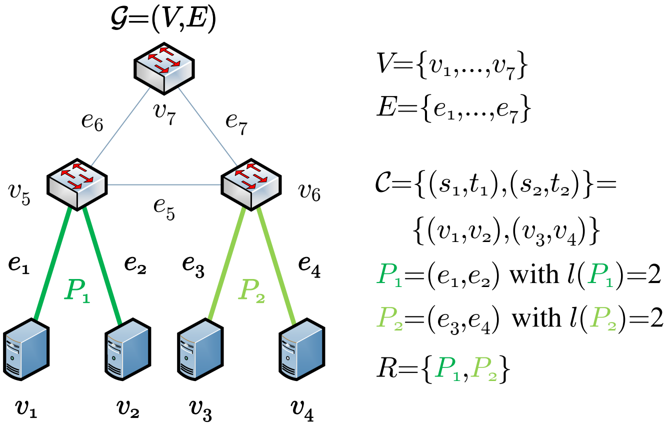


Figure 1: Example network topology including source-destination pairs and paths.

In what follows, our focus is on the feasibility of FM when the set $\mathcal{C}$ of source-destination pairs changes. Since the problem is trivial when a source-destination pair is removed, we focus on the case when a flow for a new source-destination pair is added to the existing source-destination pairs. Given a set $\mathcal{C}$ of source-destination pairs, a route $R = \{P_1, \ldots, P_N\}$ for $\mathcal{C}$, and a source destination pair (s_{N+1}, t_{N+1}), we want to find an FM $\mathcal{S}$ from route R to route $R' = \{P'_1, \ldots, P'_N\}$ such that there is a path P'_{N+1} for which $\{P'_1, \ldots, P'_N, P'_{N+1}\} \in \mathcal{R}_{\mathcal{C}'}$ is a route for the set $\mathcal{C}' = \mathcal{C} \cup \{(s_{N+1}, t_{N+1})\}$ of source-destination pairs. We refer to this as the Flow Migration Problem (FMP).

4 Flow Migration Algorithms

It is important to note that an FMP may be infeasible, either because $\mathcal{R}_{\mathcal{C}'} = \emptyset$ or because there is no valid FM to any route R'. At the same time, for a feasible FMP an FM may or may not be needed. In the simplest case of the FMP there is a path $P'_{N+1} \in \mathcal{P}_{N+1}$ such that $\{P_1, \ldots, P_N, P'_{N+1}\}$ is a route, and thus there is no FM needed. If such a path does not exist, i.e., none of the paths $P'_{N+1} \in \mathcal{P}_{N+1}$ is edge disjoint with the paths in R, then route R needs to be reconfigured into a route $R' \in \mathcal{R}^* = \{(P'_1, \ldots, P'_N) | \exists P'_{N+1} s.t. (P'_1, \ldots, P'_N, P'_{N+1}) \in \mathcal{R}_{\mathcal{C}'}\}$.

In what follows we provide two algorithms for the FMP that rely on the assumption that the set $\mathcal{R}^*$ of routes can be computed. While in general the problem of computing edge-joint paths is NP-hard, for $N = 2$ [13] provides an FPT algorithm. Furthermore, the computation of $\mathcal{R}^*$ can be feasible on certain network topologies [14], such as the FatTree topology used in Section 3.

The two algorithms we propose compute a sequence of FMs. Each FM is based on changing a single path at a time, which we call an elementary FM.

Definition 2. *Given a route* $R = \{P_1, \ldots, P_N\}$*, a route* $R' = \{P_1, \ldots, P_{i-1}, P'_i, P_{i+1}, \ldots, P_N\}$ *is an elementary FM.*

The following lemma states that any elementary FM can be composed of a sequence of atomic forwarding table updates, and is thus an FM in the sense of Definition 1.

Lemma 4.1. *An elementary FM can be performed using a sequence of atomic forwarding table updates.*

Proof. Consider an elementary FM that involves changing a path P_i into a path P'_i. If P_i and P'_i are edge disjoint then we can install forwarding table entries for P'_i starting from t_i. Otherwise, let $(v, \ldots, v')$ be a segment of P'_i that is edge disjoint with P_i. We can install forwarding table entries starting at v'. □

4.1 Direct Flow Migration (DFM)

The first algorithm we propose has polynomial complexity, but can only be used for computing a DFM, which is a permutation of the N source-destination host pairs such that switching individual paths from P_i to P'_i in the sequence using elementary FMs results in a sequence of routes.

Definition 3. *Given a route $R = \{P_1, \ldots, P_N\}$ and a route $R' = \{P'_1, \ldots, P'_N\}$, a DFM is a sequence $(i_1, \ldots, i_N)$ such that for every $1 \leq r \leq N$, $\{P'_{i_1}, \ldots, P'_{i_r}, P_{i_{r+1}}, ..., P_{i_N}\}$ is a route.*

Observe that any path in a DFM sequence may only be migrated once for the DFM algorithm to be able to find a solution and thus *at most* N elementary FMs can occur. A DFM may further not exist for arbitrary routes R and R', e.g., given $N = 2$ source-destination pairs and $P_1 = P'_2$ and $P_2 = P'_1$. In what follows, we provide an efficient algorithm for computing a direct FM if it exists. To formulate the result, let us define the directed graph $\mathcal{G}^* = (V^*, A^*)$, where $V^* = [N] = \{1, ..., N\}$, and there exists an $arc(i, j) \in A^*$ if and only if $P'_i \cap P_j \neq \emptyset$. Observe that $\mathcal{G}^*$ is in essence a conflict graph.

Lemma 4.2. *The direct FM problem has a feasible solution if and only if the graph $\mathcal{G}^*$ is acyclic. If $\mathcal{G}^*$ is acyclic, a feasible solution can be found in polynomial time.*

Proof. If $\mathcal{G}^*$ is acyclic, there must be a vertex $i_1 \in V^*$ with outdegree equal to zero, i.e., $P'_{i_1} \cap P_j = \emptyset$ for all $j \neq i_1$. Remove i_1 from the graph $\mathcal{G}^*$ and continue in the same manner to obtain the feasible sequence. On the other hand, if there exists a cycle $i_1, i_2, ..., i_{r_1}, i_r, i_1$ in $\mathcal{G}^*$, then in any feasible sequence i_1 has to appear after i_2, i_2 has to appear after i_3, etc., which is impossible. □

Thus, given routes R and R', the algorithm provided in the proof of Lemma 4.2 can be used to decide if a DFM from R to R' is possible, and to compute the sequence of elementary FMs needed. We refer to the algorithm as the DFM algorithm. Nonetheless, even though a DFM is not existent the FMP may still be feasible.

4.2 Generic Flow Migration (GFM)

Unlike the above algorithm for DFM, the GFM algorithm can migrate a path multiple times. This allows *GFM* to find Indirect Flow Migrations (IFMs) in addition to DFMs. An IFM consists of at least 3 elementary FMs, and is characterized by that at least one path is migrated more than once in the FM sequence. The pseudo-code of the GFM algorithm is shown in Algorithm 1.

Algorithm 1 GFM algorithm

```
Data: Graph $\mathcal{G}$,Routes $\mathcal{R}_\mathcal{C}$,Set $\mathcal{C}'$
Result: Flow migration sequence $\mathcal{S}$
$\forall i$ compute $\mathcal{P}_i$
Construct graph $\mathcal{T} = (\mathcal{R}_\mathcal{C}, \mathcal{F})$, where $(R, R') \in \mathcal{F}$ if $R'$ is an elementary FM of $R$
Let $\mathcal{R}^* = \{(P'_1, \ldots, P'_N) | \exists P'_{N+1} s.t. (P'_1, \ldots, P'_N, P'_{N+1}) \in \mathcal{R}_{\mathcal{C}'}\}$
Find shortest path from $R$ to any $R^* \in \mathcal{R}^*$ in graph $\mathcal{T}$
if dist$(R, R^*) < \infty$ then
    $\mathcal{S} = path(R, R^*)$
else
    $\mathcal{S} = \emptyset$
end if
```

The algorithms first computes the undirected graph $\mathcal{T} = (\mathcal{R}_\mathcal{C}, \mathcal{F})$ in which the vertices are the routes, and there is an edge between R and R' if there is an elementary FM from R and R'. Next, the algorithm computes if R is connected to any of the routes in $\mathcal{R}^*$. If it is, then the algorithm computes the shortest path between route R and any route $R' \in \mathcal{R}^*$, and returns it as the FM sequence $\mathcal{S}$. Otherwise, it returns the empty sequence. As an illustration, Figure 2 shows the graph $\mathcal{T}$ constructed for the network in Figure 1.

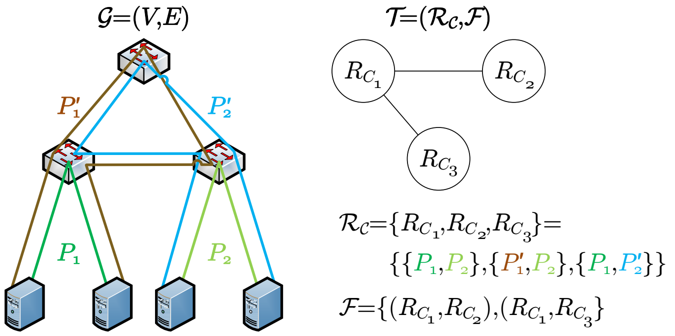

$\mathcal{G}$=(V,E)
P'_1
P'_2
P_1
P_2
$\mathcal{T}$=($\mathcal{R}_\mathcal{C}$,$\mathcal{F}$)
R_{C_1}
R_{C_2}
R_{C_3}
$\mathcal{R}_\mathcal{C}$={R_{C_1},R_{C_2},R_{C_3}}=
{{P_1,P_2},{P'_1,P_2},{P_1,P'_2}}
$\mathcal{F}$={(R_{C_1},R_{C_2}),(R_{C_1},R_{C_3}}

Figure 2: Graph $\mathcal{T}$ constructed for the network in Figure 1.

Since the graph $\mathcal{T}$ is unweighted and undirected, the complexity of computing the shortest path, and thus the complexity of GFM, is $O(|\mathcal{R}_\mathcal{C}| + |\mathcal{F}|)$ [15].

Proposition 4.3. *The GFM algorithm is correct, that is, if the FM problem has a* *solution, then the algorithm finds an FM. If the problem is infeasible, the GFM algorithm returns an empty FM sequence.*

Proof. Observe that the routes $\mathcal{R}^* \subseteq \mathcal{R}_{\mathcal{C}}$, thus both R and $R' \in \mathcal{R}^*$ are vertices of $\mathcal{T}$. Due to Lemma 4.1, there is a path from R to some $R' \in \mathcal{R}^*$, corresponding to a valid sequence of atomic forwarding table updates, if and only if the FM problem has a solution. □

4.3 Putting it all together

Upon arrival of a new flow between source-destination pair (s_{N+1}, t_{N+1}) an SDN controller would use the proposed algorithms as shown in Figure 3. If a flow for the new source-destination pair (s_{N+1}, t_{N+1}) cannot be inserted without changing the paths of the existing flows, it tries to find a Direct Flow Migration (DFM) from the initial route R to the intended route R'. If a DFM exists, the controller deploys appropriate atomic forwarding table updates (rules) on the OpenFlow switches through the SDN Southbound API. If DFM is not possible, the controller uses the GFM algorithm to find an indirect flow migration (IFM), and deploys appropriate atomic forwarding table updates (rules) on the OpenFlow switches through the SDN Southbound API.

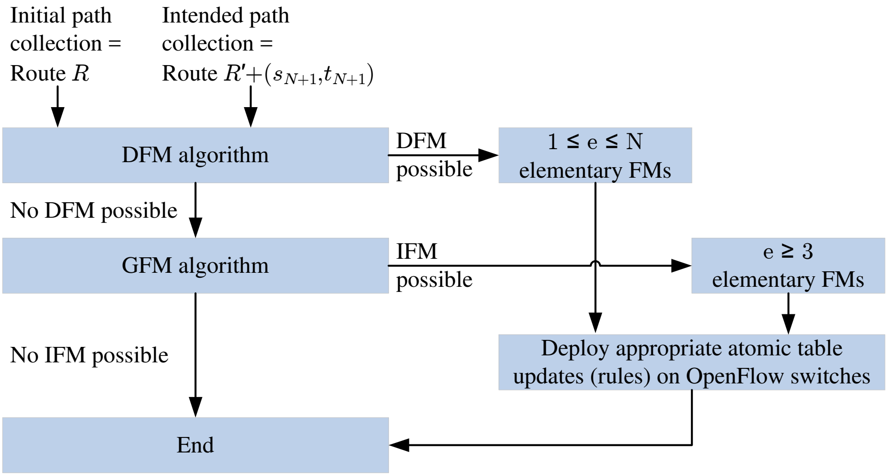
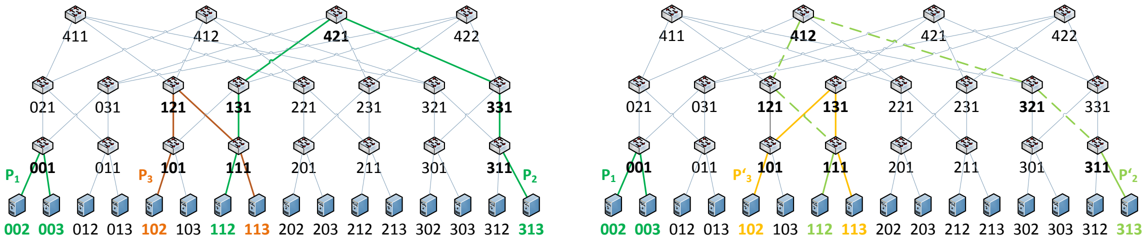

Initial path collection = Route R
Intended path collection = Route R'+(s_{N+1},t_{N+1})
DFM algorithm
DFM possible
1 ≤ e ≤ N elementary FMs
No DFM possible
GFM algorithm
IFM possible
e ≥ 3 elementary FMs
No IFM possible
Deploy appropriate atomic table updates (rules) on OpenFlow switches
End

Figure 3: Flowchart of the execution of the proposed algorithms by an SDN controller.

5 Performance Evaluation

In what follows we show numerical results to give insight into the feasibility and necessity of FM. The results were obtained in Matlab R2015b on a PC with 64 bit Windows 8, Intel i7-2600 CPU with 3.4 GHz, and 16 GB RAM. The focus of our evaluation is on the importance of flow migration, i.e., under what conditions is FM typically necessary and



(a)

(b)

Figure 4: FatTree topology with 16 hosts. (a) Path P_3 can be inserted without migrating the paths P_1 and P_2. (b) Path P_3' can only be inserted when migrating the path P_2 to P_2' through DFM. The SDN controller computing and deploying appropriate rules for FM is omitted for clarity.

how many migration steps are typically required. Our evaluation is not concerned with practical aspects of the SDN Southbound API; an implementation based evaluation is subject of our future work.

5.1 Methodology

For the evaluation we consider the FatTree topology as shown in Figure 4. Unlike the line and star topologies, used in small-scale industrial networks today, which do not provide alternative paths between hosts, the FatTree topology is a hierarchical network topology designed for efficient and resilient networking in data centers. The FatTree topology provides redundant paths between hosts, and thus it could find adoption in industrial automation environments, as it may enable the emerging standards like Time Sensitive Networking (TSN) seamless redundancy (IEEE 802.1CB [16]).

The considered FatTree topology consists of 16 hosts connected by a network consisting of 20 OpenFlow switches. The labels for hosts and switches are based on [17].

For the evaluation we considered $N = \{1, ..., 7\}$ source-destination pairs on the FatTree topology with a length constraint of 6 edges. On the FatTree topology, if each node has upward and downward degree $\leq d$, and the network has l levels, then for N source-destination pairs the set of routes has cardinality $|\mathcal{R}_{\mathcal{C}}| = (d^{2l})^N$ in the worst case.

For each N, we generated up to 5000 sets $\mathcal{C}$ of source-destination host pairs on the FatTree topology by choosing source and destination hosts at random without replacement. For each set $\mathcal{C}$ we used the APAC (All Paths And Cycle) algorithm proposed in [14] for computing all possible routes $\mathcal{R}^*$. For each set $\mathcal{C}$ and route R, we choose the $N+1$-st source-destination host pair at random, to create an instance of an FMP. Thus, for each N we consider thousands of FMPs.

As an example, for $N = 2$ a possible combination of host pairs is $\{(002, 003), (112, 313)\}$, for which a possible route, i.e., an initial collection of paths would be $R = \{P_1, P_2\}$

Table 1: Total, average, maximum, and minimum number of routes for 5000 randomly chosen sets of host pairs.

	NUMBER OF ROUTES			
N	TOTAL	AVERAGE	MINIMUM	MAXIMUM
1	18310	4	1	4
2	47617	10	1	16
3	90548	18	0	48
4	126445	25	0	144
5	116812	23	0	192
6	79572	16	0	128
7	40076	8	0	256

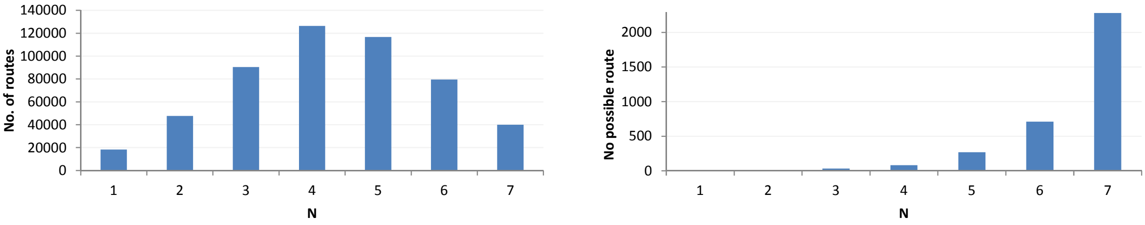
with $P_1 = (002, 001, 003)$ and $P_2 = (112, 111, 131, 421, 331, 311, 313)$ (cf. Figure 3), with path lengths $l(P_1) = 2$ and $l(P_2) = 6$, respectively. Together with a new host pair $(s_{N+1}, t_{N+1}) = (102, 113)$, these constitute an FMP. For this FMP, a path that could be inserted without flow migration would be $P_3 = (102, 101, 121, 111, 113)$ (see Figure 4 (a)), i.e., $R' = R$. However, the path $P_3' = (102, 101, 131, 111, 113)$ can only be inserted if P_2 is migrated to $P_2' = (112, 111, 121, 412, 321, 311, 313)$, which needs a direct flow migration with one elementary FM (see Figure 4 (b)), i.e., $R' = \{P_1, P_2'\} \neq R = \{P_1, P_2\}$.

Based on this evaluation set-up, we were initially interested in the number of routes (i.e. the cardinality of $\mathcal{R}^*$ for a given number of source-destination pairs) as it determines the complexity of formulating the FMP. Furthermore, we considered the number of elementary FMs needed for FM as it determines the time it takes to realize an FM, and finally the computational runtimes of solving the FMP.

5.2 Numerical Results

We start with discussing our results on the cardinality of $\mathcal{R}^*$. Figure 5 (a) shows the total number of routes over all considered sets $\mathcal{C}$ for each value of N. The figure shows that the number of routes is highest for $N = 4$, which means that the FMP is, on average, most difficult to formulate for a moderate number of flows. Table 1 shows the average, minimum, and maximum number of routes for each value N over all considered sets $\mathcal{C}$. As an example, for $N = 4$ there are on average 25 routes for each set $\mathcal{C}$ of host pairs, but there is a set $\mathcal{C}$ of host pairs with as many as 144 routes. It is interesting to note that while the average number of routes is highest for $N = 4$, the maximum number of routes increases with N. Figure 5 (b) shows the number of sets $\mathcal{C}$ for which no route exists as a function of N. The figure confirms that as N increases, for the given FatTree topology it becomes more probable that there is no route, which also limits the possibility of FM.

Necessity of flow migration: To assess the necessity of FM, Figure 6 (a) shows the share of (in)feasible FMPs for the considered instances as a function of N. We observe



(a)

(b)

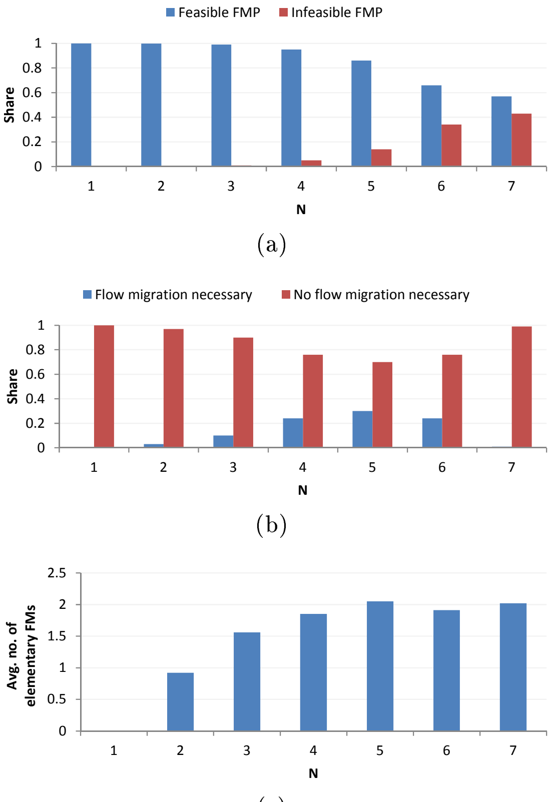
Figure 5: (a) Total number of routes over all source-destination pair sets $\mathcal{C}$ and (b) Number of host-destination pairs for which no route exists, as a function of the number N of source-destination pairs.

that the number of infeasible FMPs increases with increasing N, which is due to the fact that the probability increases that most paths in the FatTree topology are already occupied and no more path can be inserted. Furthermore, in Figure 6 (b) we show the share of FMPs in which FM is (not) necessary among the feasible FMPs (while again varying N). For low values of N, FM is typically not necessary as enough paths are left to insert an additional path. However, as we increase N the share of instances for which FM is necessary increases and peaks for $N = 5$ with a share of 30%, and decreases afterwards again. This is due to the fact that for medium-sized N an edge-disjoint path may become available if the existing paths are migrated while there is a fairly high probability that migration is possible. Nonetheless, for high N FM rarely helps, as the additional path can either be inserted without any FM or (more often) the FMP is infeasible.

Number of elementary FMs: Figure 6 (c) shows the average number of elementary FMs for those FMPs for which FM is necessary (again as a function of N) and confirms the impact of N on the number of elementary FMs. The figure shows that the number of elementary FMs is increasing up to $N = 5$ (where as many as 5 elementary FMs may be needed) but for higher N the number of elementary FMs remains constant.

Direct vs. Indirect FM: Table 2 shows the share of FMPs for which a direct FM is sufficient among the feasible FMPs that require FM, as a function of N (note that the case $N = 1$ is not shown, as an FM is never required). The results show that for $62 - 92\%$ of all feasible FMPs that need FM, a direct FM is sufficient. The importance of this result is that a direct FM can be computed in polynomial time using the DFM algorithm proposed in Section 4.1.

We can thus conclude that for lightly loaded and heavily loaded topologies FM is either not necessary or the FMP is infeasible, hence FM cannot help. Nonetheless, for moderately loaded FatTree topologies (4 to 6 flows) a significant share of the FMPs requires FM (24 % to 30 %). In those cases, roughly in 60% of the cases, the direct FM algorithm solves the FMP.



(a)

(b)

(c)

Figure 6: (a) Share of (in-)feasible FMPs, (b) share of feasible FMPs where FM is (un)necessary, and (c) average number of elementary FMs if FM is necessary, as a function of the number N of source-destination pairs.

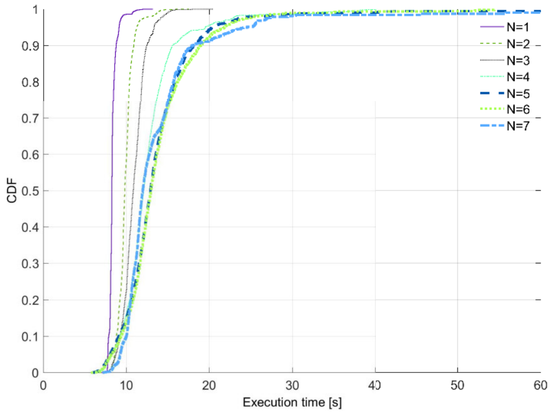


Figure 7: CDF of the execution times of the GFM algorithm.

Computation time: Finally, we evaluate the computation time needed to execute the GFM algorithm to assess the practical feasibility of FM. Table 3 shows the average time for executing the GFM algorithm (as implemented by us under Matlab) over an increasing N. The results show that the computation time increases with N, mainly due to the increasing complexity of computing $\mathcal{T}$.

To further evaluate the dependence of the computation time on N, Figure 7 shows the CDF of the execution time of the GFM algorithm, as implemented by us under Matlab. The figure reveals that as N increases the ratio of longer computation times increases (for high N we observe for instance that 10 % of the execution times are 15 s or longer).

Furthermore, while for low N almost all values are concentrated around the mean, for higher values of N the variance increases significantly. This is further evidence of the increasing complexity of determining $\mathcal{T}$ as N increases, which motivates the need for the scalable DFM algorithm as outlined in Section 4.1.

Table 2: Share of cases, in which a direct FM is possible if FM is required. $N = 1$ is excluded as an FM is not necessary in any case.

N	SHARE OF DIRECT FM CASES [%]
2	92
3	76
4	65
5	62
6	63
7	73

Table 3: Average computation time and its standard deviation for the execution of the GFM algorithm.

N	AVG. EXECUTION TIME [S]
1	8.32±0.02
2	9.89±0.02
3	10.90±0.01
4	12.67±0.02
5	13.76±0.02
6	13.87±0.02
7	14.76±0.01

6 Conclusion

In this paper, we addressed the problem of dynamic flow migration in software-defined networks. We developed two algorithms that allow conflict-free direct and indirect flow migration and thus enable inserting an additional flow at runtime. The proposed algorithm for direct flow migration runs in polynomial time but does not always find a solution to the flow migration problem. The generic algorithm finds all solutions to the flow migration problem in case it is feasible at all, but at the price of an increased computational complexity. Our results show that FM is required in 24 % up to 30 % of all cases for a FatTree topology with moderate traffic, and the necessary flow migrations typically require several migration steps. We also found that for the investigated topologies direct flow migration is possible in 62 % up to 92 % of the cases.

As part of our future work, we plan to evaluate the algorithms in a simulation environment that allows deploying atomic forwarding table updates and we will consider other topologies and compare our algorithms to existing solutions. Furthermore, we plan to relax the requirement of edge-disjointness and consider weighted graphs to allow for expressing finer-grained delay constraints.

Acknowledgment

The first author would like to thank the German Research Foundation (DFG) (research fellowship, GZ: DA 1687/2-1) for their financial support.

7 References

[1] G. P. Fettweis, “The tactile internet: Applications and challenges”, *IEEE Vehicular Technology Magazine*, vol. 9, no. 1, pp. 64–70, 2014, ISSN: 1556-6072.

[2] P. Danielis, J. Skodzik, V. Altmann, E. Schweissguth, F. Golatowski, D. Timmermann, and J. Schacht, “Survey on real-time communication via ethernet in industrial automation environments”, in *IEEE Intl. Conf. on Emerging Technology and Factory Automation (ETFA)*, Sep. 2014, pp. 1–8.

[3] J. W. Guck and W. Kellerer, “Achieving end-to-end real-time quality of service with software defined networking”, in *IEEE Intl. Conf. on Cloud Networking (CloudNet)*, Oct. 2014, pp. 70–76.

[4] E. Schweissguth, P. Danielis, C. Niemann, and D. Timmermann, “Application-aware industrial ethernet based on an sdn-supported tdma approach”, in *IEEE World Conference on Factory Communication Systems (WFCS)*, 2016, pp. 1–8.

[5] X. Jin, H. H. Liu, R. Gandhi, S. Kandula, R. Mahajan, M. Zhang, J. Rexford, and R. Wattenhofer, “Dynamic scheduling of network updates”, *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 539–550, Aug. 2014, ISSN: 0146-4833.

[6] *Software-defined networking: The new norm for networks*, White Paper, Open Networking Foundation, Apr. 2012. [Online]. Available: `https://www.opennetworking.org/images/stories/downloads/sdn-resources/white-papers/wp-sdn-newnorm.pdf`.

[7] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner, “Openflow: Enabling innovation in campus networks”, *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 2, pp. 69–74, Apr. 2008.

[8] N. Feamster, J. Rexford, and E. Zegura, “The road to sdn: An intellectual history of programmable networks”, *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 2, pp. 87–98, Apr. 2014.

[9] M. Reitblatt, N. Foster, J. Rexford, C. Schlesinger, and D. Walker, “Abstractions for network update”, in *Proc. of ACM SIGCOMM*, Helsinki, Finland, 2012, pp. 323–334, ISBN: 978-1-4503-1419-0.

[10] C.-Y. Hong, S. Kandula, R. Mahajan, M. Zhang, V. Gill, M. Nanduri, and R. Wattenhofer, “Achieving high utilization with software-driven wan”, *SIGCOMM Comput. Commun. Rev.*, vol. 43, no. 4, pp. 15–26, Aug. 2013, ISSN: 0146-4833.

[11] S. Brandt, K. T. Förster, and R. Wattenhofer, “On consistent migration of flows in sdns”, in *Proc. of IEEE Infocom*, 2016, pp. 1–9.

[12] J. Zheng, G. Chen, S. Schmid, H. Dai, and J. Wu, “Chronus: Consistent data plane updates in timed sdns”, in *2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, 2017, pp. 319–327. DOI: `10.1109/ICDCS.2017.96`.

[13] L. Cai and J. Ye, *Finding two edge-disjoint paths with length constraints*, 2015. [Online]. Available: `http://arxiv.org/abs/1509.05559`.

[14] R. Simões, "Apac: An exact algorithm for retrieving cycles and paths in all kinds of graphs", *Tékhne-Revista de Estudos Politécnicos*, no. 12, pp. 39–55, 2009.

[15] M. Thorup, "Undirected single-source shortest paths with positive integer weights in linear time", *J. ACM*, vol. 46, no. 3, pp. 362–394, May 1999, ISSN: 0004-5411. DOI: `10.1145/316542.316548`. [Online]. Available: `http://doi.acm.org/10.1145/316542.316548`.

[16] TSN Task Group, *802.1cb - frame replication and elimination for reliability*, Draft, 2017. [Online]. Available: `http://www.ieee802.org/1/pages/802.1cb.html`.

[17] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture", *SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 4, pp. 63–74, Aug. 2008, ISSN: 0146-4833.

Paper E

Integration of QoS Parameters from IEEE 802.11s WLAN Mesh Networks into Logical P2P Overlays

Michael Rethfeldt, Peter Danielis, Björn Konieczek, Felix Uster, Dirk Timmermann

In Proceedings of the 14th IEEE International Conference on Ubiquitous Computing and Communications (IUCC), pp. 1170-1177,
`https://doi.org/10.1109/CIT/IUCC/DASC/PICOM.2015.175`*, ISBN: 978-1-5090-0153-8, Liverpool, England, Oktober 2015*

Integration of QoS Parameters from IEEE 802.11s WLAN Mesh Networks into Logical P2P Overlays

Michael Rethfeldt, Peter Danielis, Björn Konieczek, Felix Uster, Dirk Timmermann,
University of Rostock
Institute of Applied Microelectronics and Computer Engineering
18051 Rostock, Germany, Tel.: +49 381 498-7269
Email: michael.rethfeldt@uni-rostock.de

Abstract

Adopted in late 2011, IEEE 802.11s comes as the first industry standard to enable vendor-independent and inter-operable WLAN mesh networks. Featuring automatic device interconnection and routing, they provide a higher scalability, flexibility, and robustness compared to common centralized WLAN infrastructures. The 802.11s standard defines mandatory support of the Hybrid Wireless Mesh Protocol (HWMP) and Airtime Link Metric (ALM) for MAC-layer routing. While 802.11s covers the physical network underlay, Peer-to-Peer (P2P) protocols are an equivalent on the application level. In contrast to centralized client/server communication, they establish a fail-safe and scalable logical network overlay for, e.g., distributed content sharing, streaming, search, or synchronization. Thus, P2P networks exhibit many of the characteristics of physical WLAN mesh networks. It is obvious to consider joint solutions, where both technologies are combined to leverage future distributed local area wireless applications. Nevertheless, common P2P protocols, such as BitTorrent, do not consider the structure of the physical underlay while performing topology management. Furthermore, they are primarily designed to be used over wired communication networks such as large parts of the Internet. When deployed over WLAN mesh networks with their quickly varying channel conditions, BitTorrent shows severe performance drawbacks. We present a cross-layer approach based on 802.11s and BitTorrent, that optimizes application layer peer selection by considering the mesh standard's routing metric ALM. Our solution was implemented and evaluated in a real-world test bed. Results show that average download time can be reduced by up to 20 % already in small network setups.

1 Introduction

Endorsed by the increasing variety and affordability of wireless consumer devices, complex networks can be established to provide distributed, content-centric services in places of high node density and mobility. The widespread IEEE 802.11 WLAN (Wireless Local

Area Network) standard family [1] is already omnipresent in today's home networks, office environments, and public facilities. In contrast to currently prevalent "infrastructure" deployments based on central Access Points (AP), decentralized WLAN mesh networks are characterized by their flexible, scalable, and fail-safe network topology [2]. The standard amendment IEEE 802.11s was adopted in late 2011 and initially adds mesh functionality to the WLAN MAC layer. In an 802.11s network, every mesh node provides data forwarding and routing capabilities [3]. Nodes within radio range automatically interconnect and establish paths to selected targets. Economic network extension is simply possible by bringing in additional mesh nodes. Thereby, path maintenance follows a radio-aware link metric that takes the properties of the wireless medium and MAC protocol into account. Thus, the network becomes more robust to changes in node availability, density, and varying link qualities. For interoperability, 802.11s defines the *Hybrid Wireless Mesh Protocol* (HWMP) and the *Airtime Link Metric* (ALM) as default combination for path selection [4].

P2P technology is used to develop distributed, scalable, and failure-resilient network applications. It overcomes the drawbacks of centralized client/server communication, that inherently includes a Single Point of Failure (SPoF) [5]. Possible applications include multimedia streaming, Voice over IP, content search, synchronization, or file sharing. P2P networks are implemented as logical overlay networks on top of a given physical underlay. Regarding robustness and scalability, they share many similarities with wireless mesh networks [6]. Therefore, logical P2P networks are particularly suitable as candidate application above a mesh underlay. A combined solution features scalability and robustness both on logical application and physical network level [7]. In a WLAN mesh network, a P2P overlay could be used to distribute firmware and configuration updates or network statistics for management, or to implement live streaming as application-layer multicast.

However, common P2P file sharing protocols, such as the well-known and widespread BitTorrent (BT) [8], are designed to be used over the Internet. Thus, logical topology management is optimized for wired networks, i.e. reliable links and stable channel conditions. On the contrary, link quality may change quickly in wireless mesh networks. Neighboring devices within radio range interfere on the same channel and suffer from contention-based medium access overhead. While nodes on P2P application level always appear as single-hop neighbors, they may be multiple hops away in the physical mesh underlay. Application-layer traffic over multi-hop paths puts stress on intermediary nodes that only forward data to an overlay endpoint. These nodes may also be interested in the forwarded data, but are not optimally chosen as destination in the overlay. This situation induces redundant data transmission and channel access, compared to a strategy where overlay traffic is kept physically local. Thus, when deployed over WLAN mesh networks, the default BT protocol reveals serious performance drawbacks, when not considering the physical network topology during peer selection [9].

Consequently, we have developed a cross-layer solution to overcome the mismatch between a logical P2P overlay based on BitTorrent and a physical WLAN mesh underlay based on IEEE 802.11s. We modified the default BitTorrent choking algorithm to use the MAC-layer Airtime Link Metric (ALM) of 802.11s as criterion for peer selection. Comparing different underlay technologies, we show the need for BitTorrent optimization. Measurement results for the default and the modified choking algorithm show, that average download time can be reduced by up to 20 %, already by directly applying ALM for peer selection and combining it with a limited neighborhood scope.

The remainder of this paper is organized as follows: Section 2 first outlines the basic principles of the IEEE 802.11s WLAN mesh standard and its Linux implementation *open80211s*. Then, we introduce the BitTorrent P2P file sharing protocol and its peer selection algorithm. Finally, we point out the mismatch between logical P2P overlay and physical mesh underlay. In Section 3, we discuss related work in combining WLAN Mesh Networks and BitTorrent. Section 4 illustrates the design of our cross-layer solution and the modifications we applied to the default BitTorrent peer selection. In Section 3, we discuss the measurement results that were obtained in a real-world test bed. Finally, we give a conclusion in Section 7 and briefly state possible improvements and approaches for future research.

2 Technological basis

2.1 IEEE 802.11s

As first common industry WLAN mesh standard, IEEE 802.11s was ratified in September 2011 [1, 10, 3]. It enables vendor-independent infrastructure-less multi-hop communication based on the widespread WLAN technology. The central Access Point (AP) role is delegated to all distributed nodes, as each Mesh Point (MP) in an 802.11s network supports frame forwarding and path selection. Mesh functionality is directly integrated into the 802.11 MAC layer specification. Any changes to the underlying physical layer are avoided and mesh support can be easily added on the driver level to be used with existing WLAN hardware. The common 802.11 data and management frames have been extended to enable automatic peer discovery, peer link establishment, frame forwarding and routing. Figure 1 shows 802.11s in the ISO/OSI stack. Since path selection is handled on the MAC layer, it is supposed to generate less overhead than existing network-layer mesh routing protocols [10, 11]. Moreover, mesh operation becomes transparent to all higher layers.

To ensure interoperability, every MP must support the *Hybrid Wireless Mesh Protocol* (HWMP) and the *Airtime Link Metric* (ALM) [4] as mandatory default combination for path selection. HWMP is based on the reactive Ad-Hoc On-Demand Distance Vector (AODV) routing protocol [12]. Analogous to AODV's route request (RREQ) and reply (RREP) messages, HWMP defines path request and reply frames (PREQ/PREP). Optionally, a tree-based proactive routing mode is available. This is designated for paths to

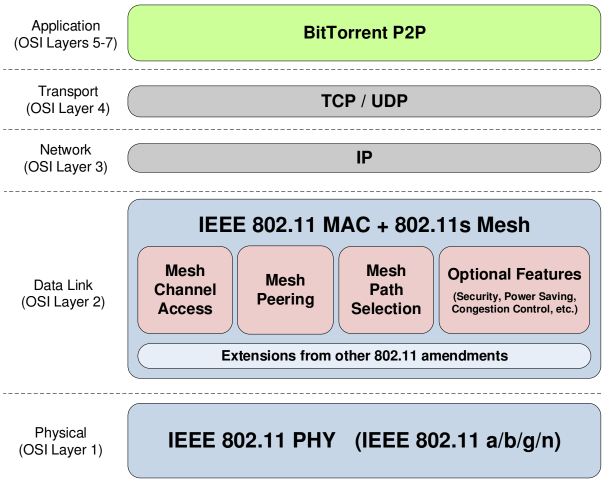


Figure 1: IEEE 802.11s in the ISO/OSI stack

static mesh nodes such as gateways and may be used along with the reactive on-demand mode [10, 13]. If a source MP needs to establish a path to a destination MP on-demand, it broadcasts path requests (PREQ) to all nodes within radio range. Each intermediary MP along a possible path adds the ALM value of the receiving link to the metric field in the PREQ frame and forwards it. Only PREQs with better (smaller) ALM value and higher sequence number are forwarded. Once the destination is found, a PREP is send as unicast along the best path, found in the PREQ flooding phase. When the PREP finally reaches its PREQ source, the bidirectional path is established.

Mesh paths are chosen according to the link metric, which is ALM by default. It represents the costs for transmitting a frame over a specific link in the mesh network by considering technology parameters of the WLAN physical layer and the wireless medium. The so-called *airtime cost* (c_a) is calculated as follows:

$$c_a = \left[O_{ca} + O_p + \frac{B_t}{r} \right] \cdot \frac{1}{1 - e_{fr}} \tag{1}$$

O_{ca} and O_p are constants for the channel access and MAC protocol overhead. B_t is the test frame size. By default, a frame size of 8192 bit is used. r denotes the test frame data rate, given in Mbit/s, whereas e_{fr} denotes the expected frame error rate. The estimation of e_{fr} as well as the values of the overhead constants are not predefined by the 802.11s standard but left open to vendor implementations [14].

As usual for distance vector protocols, an MP using HWMP only knows its direct neighbors and nodes in multi-hop distance, to which communication has been explicitly initiated, e.g., on higher layers. Path tables on every MP contain forwarding rules to target nodes via neighbors ("next hops"). These rules are periodically updated in small intervals within their expiry time. To every target, only the best next hop with smallest ALM is kept. Since a path commonly consists of multiple consecutive links, ALM works as cumulative metric. Although shorter paths often result in a smaller metric, longer paths may sometimes be preferred. This occurs if intermediate nodes on a longer path have better peer links in total (higher data rate and/or less frame errors) and thus the cumulative path cost is smaller than that of the alternative shorter path. Thus, ALM represents both overall link quality and path length. The studies in [15] and [16] show that the 802.11s mandatory default combination of HWMP and ALM outperforms other routing protocol combinations with the metrics ETX and ETT in different traffic and network scenarios.

2.2 open80211s

The Linux project *open80211s* [17] is currently the most advanced open-source reference implementation of 802.11s. It already satisfies all mandatory and various optional parts of the standard. The code base is integral part of the *mac80211* kernel module, i.e. the software MAC layer of the Linux WLAN stack [18]. Since some parameters in ALM calculation are left open to vendor implementations, open80211s provides own variants for frame error rate estimation and overhead constants (see equation 1). While O_{ca} and O_p are summarized to 1, the data rate r (of the last unicast frame transmission) depends on the rate control algorithm (RCA). In current Linux kernels, *minstrel* is used as default RCA [19]. The estimation of e_f follows equation 2:

$$e_f\left[t_k\right] = \frac{80 \cdot e_f\left[t_{k-1}\right] + 5}{100} + 20 \cdot \delta\left[t_k\right] \tag{2}$$

The Boolean parameter $\delta\left[t_k\right]$ indicates a successful (0) or failed (1) last frame transmission. The error rate results in values between 0 and 100 (the right operand is truncated). It is then normalized to range from 0 to 1, before being used in the final ALM calculation (equation 1). As e_f is updated on every frame transmission, always the most recent value is available on a triggered path refresh. Real-world evaluations have already demonstrated the HWMP performance of open80211s [20, 14].

2.3 BitTorrent

Currently, BitTorrent (BT) is the most prevalent P2P network for file sharing. It contributes about 6 % to the overall Internet traffic [21]. Its popularity results from its capability of achieving high download rates, which is usually the main interest of users.

For each file to be shared, one specific logical network is created. To search for a file, usually a web site is contacted to get a *.torrent* metadata file. This file contains, among other things, the address of a *tracker* and information about the file to be downloaded. The tracker is contacted to get a list of BT users (peers) holding the file (or parts of it – so-called *pieces*, which are further subdivided into *sub-pieces*). Thereby, the pieces and sub-pieces can be of different size, e.g., ranging from 32 kBytes to 32 MBytes. All peers, which are interested in this file, form a so-called *swarm*. Complete downloaders serving the whole file are called *seeds*. Incomplete downloaders are called *leechers*. BT peers start to download pieces in *random* order and change to *rarest first* order after the first piece is completed [22]. Thereby, they follow the *strict priority* of solely requesting sub-pieces of a particular piece before sub-pieces from the next piece.

For selecting other peers who may download a piece, each peer applies the so-called *choking algorithm* [22]. In a nutshell, this is a variant of the tit-for-tat strategy. Only peers offering sufficient upload performance are given download time in return (they are *unchoked*). The choking algorithm to determine a peer that may download pieces is executed periodically because upload performance of peers can change quickly. As an exception, each peer has an *optimistic unchoke* available to unchoke one other peer regardless of his upload performance. This is performed to increase piece diversity and parallelism of data transmissions in the network. By that, faraway peers can also contribute in redistribution of pieces, although they would not have been chosen following the upload rate criterion. By default, the choking algorithm is executed in a 10 seconds period. Given the number of upload slots N (4 by default), N-1 peers are unchoked as a result. Every 30 seconds a new optimistic unchoke is selected for the remaining upload slot.

Once a peer has finished its download, it may decide to stay in the network for a while (lingering), operating as a seed. During this time span, it only uploads pieces preferring peers, to which it has the best upload rates.

2.4 Mismatch between P2P overlay and mesh underlay

Common P2P file sharing protocols, such as BitTorrent (BT), do not consider the structure of the physical underlay while performing topology management and peer selection. They are optimized for wired networks, which show reliable links and stable channel conditions. In wireless mesh networks, however, link conditions may vary quickly due to distance, mobility, or the occurrence of obstacles, obscuring the line of sight between nodes. Devices within radio range interfere on the same channel and suffer from contention-based medium access overhead. Moreover, in a multi-hop scenario the cost of a transmission increases, as each forwarding step requires additional channel access and is again subject to possible frame errors and collisions. Therefore, deploying BT over a wireless mesh underlay is challenging and the selection of suitable logical peers has an immense effect on network performance.

While BT peers on application level always appear as direct neighbors, they may be

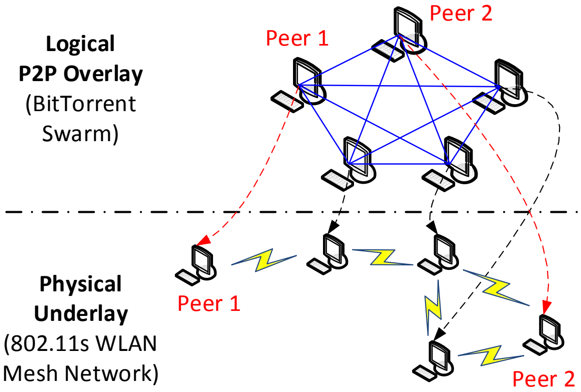


Figure 2: Mismatch between logical overlay and physical underlay

multiple hops away from each other in the physical mesh underlay and wireless link quality can differ severely. In Figure 2, this is shown for Peer 1 and 2, which are connected by a three-hop mesh path in the underlay but maintained as immediate neighbors in the BT overlay. Imagining the intention of content synchronization starting from Peer 1 as a seed, selecting Peer 2 as a leecher would be the worst case, as this generates unnecessary stress on the intermediary nodes. Instead, physically close peers should be usually preferred, which allows for collaborative distribution of pieces while traffic is kept local. Moreover, peers that are endpoints of low-quality paths should be avoided and saved up to be served by more suitable peers in the distribution process. Both aspects can be faced by the 802.11s standard's routing metric ALM (see Section 2.1), as it expresses both path length and overall link quality. Thus, it is a better criterion for BT peer selection in 802.11s mesh networks compared to the default BT choking algorithm.

3 Related Work

There are already some approaches, which intend to optimize the performance of the BitTorrent (BT) protocol when applied over mobile ad hoc networks.

In [23], a cross-layer solution is presented whereby the BT tracker receives data from the IETF Application Layer Traffic Optimization (ALTO) infrastructure in the form of distance information of the Open Link State Routing (OLSR) proactive mesh routing protocol. As a result, the tracker returns a list of peers sorted after this distance information rather than returning random peers. However, the metric used for the cost of links is the Expected Transmission Count (ETX), which solely minimizes the expected total number of packet transmissions to successfully deliver a packet to the destination. Instead, we provide BT with information in the form of the ALM metric, which allows for inclusion of WLAN technology-specific parameters and thus is a better measure for the link quality. Moreover, as support of HWMP and ALM were defined as mandatory in the 802.11s standard, ALM will always be available on each compliant node, in contrast to other metrics.

The BitHoc project aims at adapting BT to wireless ad hoc networks mainly by restricting the communication to peers, which are only some hops away from each other. The authors suggest to realize the neighborhood scope restriction by using the TTL value for reducing routing overhead [24]. The results show that the overall download time can be reduced and the throughput can be improved when using a modified choking algorithm and piece selection strategy in combination with restricting neighborhood scope. Thereby, the authors still allow optimistic unchokes so that not only adjacent peers get the chance to download but also a few remote peers. However, the ALM metric represents a better quality metric than only the number of hops, as multi-hop paths may occasionally be preferable compared to overloaded one- or two-hop paths.

The work described in [25] is one of the few publications, which apply a P2P network on top of an 802.11s mesh network, and discuss possible performance optimizations by using cross-layer approaches. Thereby, the ALM metric is passed to a generic software framework that can be used as API for P2P applications. However, no specific P2P protocol, such as BT, is evaluated and no effect of ALM utilization is measured. Moreover, the 802.11s routing protocol HWMP is required to be heavily extended on each peer, e.g. by vendor-specific frame fields. On the contrary, we want to avoid MAC-layer modifications to ensure interoperability with standard 802.11s.

The work in [26] proposes a distributed management protocol for mobile P2P networks to, e.g., replace the centralized tracker in BT. As a result, the peers organize themselves in a shared tree dedicated for disseminating membership information. By using ad hoc routing information, peers construct and adapt their logical links in the tree with regard to the current network topology. However, this requires adding structure to unstructured P2P networks following routing characteristics. Furthermore, traffic overhead is created for maintaining this structure. Instead, in our approach the BT protocol is only slightly modified to select proximate peers by means of a new choking algorithm. Thereby, no modification of the construction algorithm is necessary.

4 Cross-Layer Approach

As depicted in Section 2.4, applying P2P protocols such as BT over WLAN mesh networks reveals a mismatch between logical overlay and physical underlay. To address this, we follow a cross-layer approach that performs an integration of 802.11s HWMP information (links and paths with ALM) into the BT protocol and its choking algorithm. As a consequence, the physical mesh topology is considered during logical peer selection. Figure 3 shows the design of our cross-layer solution. The software components *Mesh Management Framework* and *BitTorrent Client* are explained in the following.

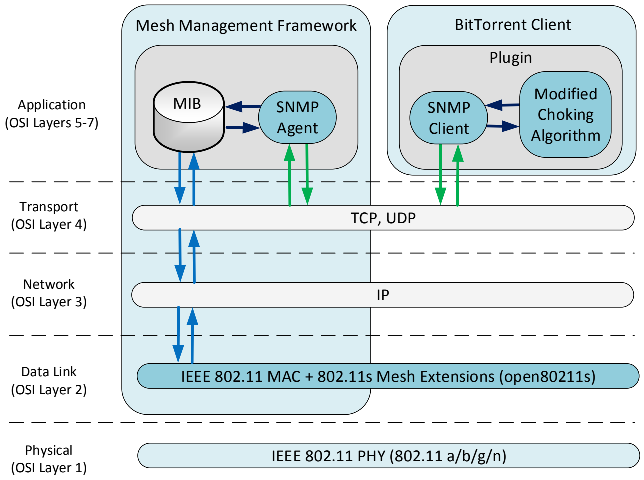

Mesh Management Framework
BitTorrent Client
Plugin
MIB
SNMP Agent
SNMP Client
Modified Choking Algorithm
Application (OSI Layers 5-7)
Transport (OSI Layer 4)
TCP, UDP
Network (OSI Layer 3)
IP
Data Link (OSI Layer 2)
IEEE 802.11 MAC + 802.11s Mesh Extensions (open80211s)
Physical (OSI Layer 1)
IEEE 802.11 PHY (802.11 a/b/g/n)

Figure 3: Proposed cross-layer approach

4.1 Mesh Management Framework

The first part of our solution is based on a management framework for 802.11s mesh networks that we developed at the University of Rostock [27]. It was written in Java for flexible deployment on heterogeneous platforms. Running on each mesh node, the framework encapsulates the features of the Linux kernel-level 802.11s implementation *open80211s* (see Section 2.2) and several CLI tools in its platform-dependent part, which is interchangeable for possible future 802.11s implementations. In a platform-independent part, status information and configuration functions are represented by Java classes. On this abstraction level, our framework addresses automatic mesh network initialization. The bootstrapping process includes MAC- and IP-layer auto-configuration to ensure node availability and to enable communication between overlying applications. Each node is equipped with fall-back procedures to repeat certain configuration steps on occurrence of errors. Furthermore, every node runs an integrated agent for SNMP (Simple Network Management Protocol), a widespread industry network management protocol. Local status information and configuration functions are provided as SNMP data model and can be queried by a corresponding SNMP client. Thus, SNMP serves as interface to easily manage the mesh nodes remotely and to combine their otherwise limited network view (see Section 2.1) to a global scope. Our framework further differentiates between an *Agent* and *Manager* role. The latter additionally supports the complementary SNMP client side, including the remote query of status information and configuration of nodes. It also provides a DHCP server for IP address distribution and NTP server for time synchronization.

The Linux kernel 802.11s implementation on every node maintains data structures for link establishment and HWMP path selection [17]. The *station list* and *mesh path list* contain all physical links and logical paths from a mesh node's viewpoint. While every station list entry holds the information of a direct link to a node's one-hop neighbor, any forwarding rule in the mesh path list includes next hop, target node, and ALM value. When using default settings, path information are updated every second. A path expires after five seconds, e.g., path establishment is re-triggered. These information are reflected as SNMP objects in our management framework. They can be queried either by an external or any local application, as long as it supports SNMP client functionality. Thus, SNMP is used as an interface to pass the MAC-layer link and path information, gathered by the mesh management framework, to a BT client application, running on the same machine (see Figure 3).

4.2 BitTorrent Client and Plugin

The open source software *Vuze* (former Azureus) is one of the most popular BT clients available [28]. It is written in Java and can be easily extended, as it provides a flexible plugin API. We used a legacy program version (Vuze 4.3.1.4) to be compatible with an existing open source plugin that served as template for integrating our peer selection optimization. The Vuze plugin called "BASS" originates from an external research project [29]. Its vanilla version already includes dummy code to replace the default peer selection of Vuze. However, the initial BASS plugin state only performs logging functionality and makes no changes to the BT choking algorithm. We integrated our modifications into the plugin skeleton code. When the plugin is loaded, it replaces the default *Unchoker* implementation of Vuze. To access the physical mesh link and path information inside the local mesh management framework on a node, we integrated an SNMP client into the plugin. Up-to-date mesh status information are queried periodically every 8 seconds, to be available each time the BT choking algorithm is triggered (every 10 seconds).

4.3 Modified Choking Algorithm

As depicted in Section 2.3 the choking algorithm handles BT peer selection. The algorithm periodically grants a predefined number of upload slots to interested peers. We replaced BT's default unchoking criterion, the upload rate of peers, by the ALM metric of the mesh path to it. ALM (see Section 2.1, equation 1) represents the frame transmission time cost in microseconds. Thus, smaller metric values are preferred and the list of interested BT peers is now sorted in ascending order. According to equation 1, ALM considers parameters of the wireless channel as well as the current frame transmission and error rate. Its cumulative nature also reflects the path length, but additionally enables the consideration of fast multi-hop and overloaded single-hop paths. Thus, we assume an improvement in overall download performance already when directly applying ALM as unchoking criterion in an 802.11s mesh network. Especially in larger mesh setups with many multi-hop paths, this approach is expected to keep network strain local by choosing proximate peers to upload pieces to.

Apart from that, we did not change the default timing of the BT choking algorithm (10 s and 30 s for optimistic unchokes, respectively) and number of upload slots ($4 = 3 + 1$

Table 1: Vuze BitTorrent settings

Parameter	Value
File size	65 MB
Piece size	256 kB
Unchoke period	10 s
Optimistic unchoke period	30 s
Upload slots	4 (3+1)

optimistic unchoke) [22]. The fourth slot for the randomly chosen optimistic unchoke is explicitly left untouched because it facilitates piece diversity and parallelism in the BT swarm, as stated in [24].

In smaller mesh setups with only few and rather short multi-hop paths, peer proximity and robustness to small metric deviations between different paths should be further enforced within the choking algorithm. Consequently, we added the option to limit possible peers to only those in one-hop distance, while still ordering them by ALM. This filtering step can be performed on every node. In fact, each mesh node can differentiate between direct neighbors (contained both in *station list* and *mesh path list*) and nodes in multi-hop distance (only contained in the *mesh path list*) out-of-the-box, as given by the 802.11s standard's default path selection protocol HWMP. This already enables one-hop limitation solely by relying on standard features. Complete path knowledge for a more fine-grained hop count filtering would require aggregation of all distributed path information on every node. This was not considered due to the implied network overhead.

For evaluation of the three choking algorithm variants (default BT, ALM-based, ALM-based & limited to single-hop peers), we integrated them in the Vuze plugin code and made them switchable by an external configuration file.

5 Evaluation in a Real-World Test Bed

We established a real-world test bed with 8 nodes to realistically evaluate our solution. It comprised a notebook (1.6 GHz Core i5-4200U dual-core CPU, 8 GB RAM, Ubuntu 14.04 LTS, Kernel v3.13) and 7 Raspberry Pi (RPi) model B single-board computers (700 MHz ARMv6 CPU, 512 MB RAM, Raspbian Linux, Kernel v3.12) [30]. Each node was equipped with an USB WLAN adapter (Buffalo WLI-UC-GNM), operating in the 2.4 GHz ISM band in 802.11g mode. The adapters rely on the Ralink chipset driver *rt2800usb* that supports the open80211s mesh extensions [31, 32].

A measurement run consisted of the distribution of a 65 MB video file to all nodes in the BT swarm, which is a sufficient size to show the effect of our modifications to the BT choking algorithm. Measurements were performed for each of the three choking algorithm variants (see Section 4.3). Thus, the modified Vuze client and BASS plugin (see Section

Table 2: open80211s HWMP settings

Parameter	Value
HWMP frame TTL	31
Max. PREQ retries	4
Path refresh time	1000 ms
Min. discovery timeout	100 ms
Active path timeout	5000 ms
PREQ min. interval	10 ms
HWMP net diameter traversal time	50 ms

4.2) were installed on every device. The notebook always operated as initial BT seed and tracker, the RPis as distributed leechers. Every leecher was configured to become a seed as soon as it has received all pieces of the file. The lingering time was set to its maximum value to ensure a seed remains in the BT swarm for the complete measurement duration.

Measurements were initially performed comparatively on different underlays (see Section 5.1). In a second step, a multi-hop mesh setup was used (see Section 5.2). The 802.11s mesh operation and bootstrapping, including IP address configuration, were performed by our management framework running on each node (Manager role on the notebook, Agent role on the RPis). Table 1 shows the basic BT parameters that were used for all measurements. Apart from that, no upload/download speed limitations were defined on BT client side. Table 2 shows the HWMP related open80211s settings that were used in the mesh setups.

Additionally, *tshark*, a CLI version of *wireshark*, was executed on all nodes to capture incoming and outgoing TCP packets for BT traffic analysis [33]. After each completed file distribution to all peers in the BT swarm, the Vuze log files and traffic captures of all nodes were collected. Using these, the required time to receive the whole file was determined for every peer.

5.1 Comparison of different underlay technologies

Initial measurements were performed on different underlays to retrieve reference times for the achievable BT performance, depending on the kind of physical LAN setup. First, all peers were connected by a Fast Ethernet switch (D-Link DES-1008D, 10/100 Mbit/s gross bandwidth). In a second setup, the switch was replaced by a WLAN AP (ASUS WL-500gP, 54 Mbit/s gross bandwidth in 802.11g mode). Here, all nodes were placed within 1 m distance to the AP, their WLAN adapters configured in common infrastructure mode. Naturally, only the unmodified BT choking algorithm was used in these first two setups. As our optimized choking alternative relies on ALM, it is only applicable in a mesh underlay.

Finally, a small 802.11s mesh network was set up. All nodes were still placed as in the AP case, i.e., within direct radio range to each other, to cover the same area as the former WLAN infrastructure. Due to the small inter-node distance, only single-hop mesh paths were established with little divergence in ALM values. In the case of the single-hop mesh network, measurements were performed both for the default BT choking algorithm and the optimized variant with ALM as peer sorting criterion.

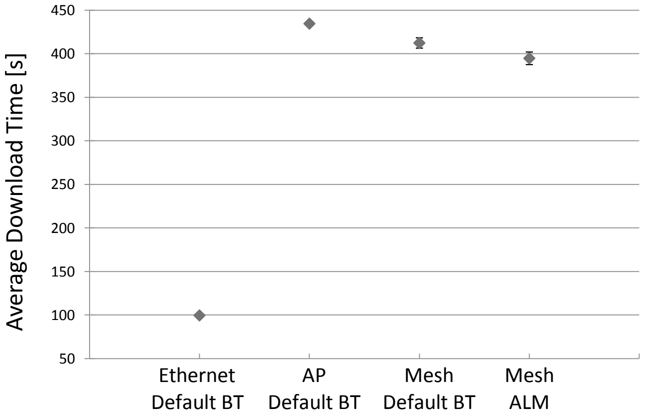


Figure 4: Average download time for different underlay networks

For any of the described combinations of underlay and choking algorithm, 5 measurement runs were performed and the required download time was averaged over all peers. Figure 4 displays the results, including the 95 % confidence intervals. In the Ethernet setup, it took an average of 100 s on each node to download the 65 MB file. As expected due to lower net bandwidth and additional wireless channel access overhead, download time increased for the AP setup to an average of 435 s, i.e. it was more than 4 times slower than in the wired network. Subsequently replacing the WLAN infrastructure by an 802.11s mesh network without changing the node placement, download time improved again by 5 % to an average of 412 s. This reveals the competitiveness of 802.11s to the likewise range-limited, AP-based setup. Eventually, using plain ALM as criterion for peer selection already resulted in further improvement by 4 % to an average of 395 s, even for the mere presence of one-hop paths with small link quality variations that are otherwise not considered by the default BT algorithm.

5.2 Measurements in a multi-hop mesh setup

Consequently, as this is the common usage scenario for WLAN mesh networks, we performed similar measurements in an 802.11s setup of wider network diameter, that is not coverable by a single AP anymore. Nodes were placed in different rooms spread over two floors of our institute building, as depicted in Figure 5. The higher inter-node distance and signal loss due to walls and objects lead to considerable ALM fluctuations and the establishment of multi-hop paths. On the one hand, this setup shows the flexibility of extending the coverable area of the wireless backbone merely by node placement. On the other hand, the delay caused by frame forwarding on intermediary nodes and the varying link quality also imply a degradation in network performance.

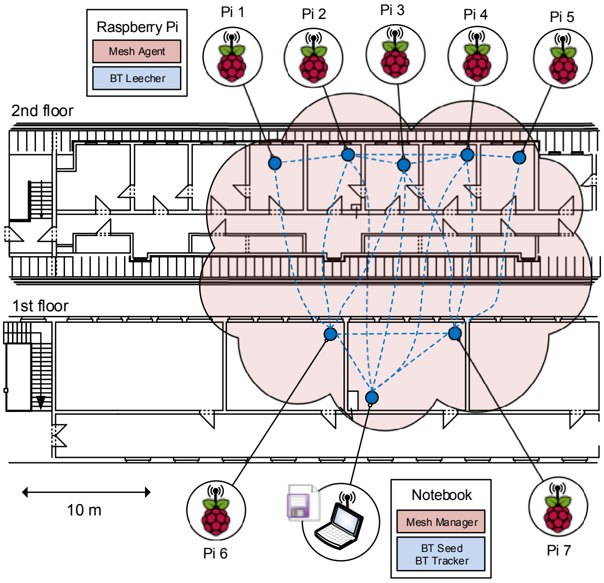


Figure 5: Floor plan of the multi-hop mesh setup

We evaluated the three choking algorithm variants described in Section 4.3: default BT mode (1); peers ordered by ALM (2); only one-hop peers allowed, ordered by ALM (3). Again, 5 measurement runs were performed for each variant and the required download time was averaged over all peers. Results are displayed in Figure 6.

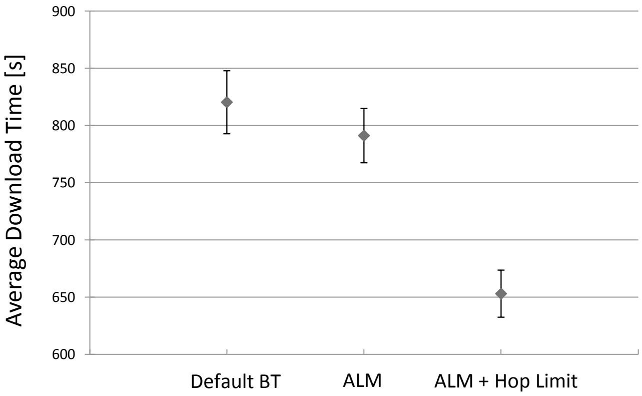


Figure 6: Average download time in the multi-hop mesh setup

In the multi-hop setup, we could clearly observe the need for optimization of the default BT choking algorithm. As expected, average download time severely increased from 412 s (default BT mode, single-hop mesh) to an average of 820 s (default BT mode, multi-hop mesh), i.e. it nearly doubled. By applying ALM as peer sorting criterion, required time decreased again by around 4 % to an average of 791 s. Additionally limiting upload only to peers in one-hop distance, still ordered by ALM, yielded an average download time of 653 s. This means a reduction of around 20 % compared to running the default choking algorithm in this setup.

Thus, the approach of keeping BT traffic local leads to considerable improvement already for a small mesh network size of only 8 nodes and average paths lengths of 2 hops. As ALM expresses both length and overall link quality of a path, BT peers are chosen with regard the topology and quality of the mesh underlay. This reduces stress on intermediary nodes and redundant piece transmissions (see Section 2.4).

Our optimization relies solely on standard features provided by 802.11s, i.e., it uses plain ALM and conducts one-hop limitation simply by comparing the list of direct neighbors with a path's target node. We expect this to perform even better in large-scale mesh setups with higher node count and longer paths. Transmission over many hops is very costly due to the required data forwarding on each intermediary node (see Section 2.4) and thus peer selection has an even greater effect on BT performance.

6 Conclusion

In the presented work, we use our mesh management framework as a middle-ware for exploring cross-layer optimization strategies within the specification boundaries of the 802.11s standard. We pursue a bottom-up approach that performs integration of the 802.11s default MAC-layer Airtime Link Metric (ALM) into the application layer to improve the BitTorrent (BT) P2P protocol and its choking algorithm, running on top of the physical mesh network. Defined as mandatory default metric in 802.11s, ALM is guaranteed to be available on every standard-compliant mesh node. To the best of our knowledge, the presented cross-layer solution is the first approach to integrate ALM into BT while no changes to the 802.11s routing protocol HWMP are required. As a logical starting point, we pass plain ALM to BT's choking algorithm to replace the upload rate as default peer selection criterion. Directly using ALM shows improvement already in our small mesh setup and average file download time on each node is reduced by around 4 %. Consequently, we combine ALM with a neighborhood scope limit. In our test bed, the second approach reduces average download time by up to 20 %, by still maintaining interoperability to the 802.11s standard and HWMP. The presented mesh setup represents a decisive step towards the mitigation of the mismatch between logical overlays and wireless physical underlays and hence forms the basis for establishing an even more comprehensive test bed. The trend of achievable download time reduction is clearly visible using this setup and fixed parameter set (BitTorrent defaults). In our future work, we will investigate our solution in larger and more dynamic scenarios. Both a 40 node real-world test bed and a simulation environment are currently prepared. We will evaluate the influence of varying parameters, such as choking period, number of upload slots, and piece size.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG), Research Training Group 1424 (Multimodal Smart Appliance Ensembles for Mobile Applications) for their financial support.

7 References

[1] "IEEE Standard for Information technology - Telecommunications and information exchange between Systems - Local and metropolitan area networks - Specific requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", *IEEE Std 802.11-2012 (Revision of IEEE Std 802.11-2007)*, 2012.

[2] M. Portmann and A. A. Pirzada, "Wireless mesh networks for public safety and crisis management applications", *Internet Computing, IEEE*, vol. 12, no. 1, pp. 18–25, 2008.

[3] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "IEEE 802.11s: The WLAN Mesh Standard", *Wireless Commun.*, vol. 17, no. 1, pp. 104–111, Feb. 2010.

[4] S. Bari, F Anwar, and M. Masud, "Performance study of hybrid wireless mesh protocol (HWMP) for IEEE 802.11s WLAN mesh networks", in *Computer and Communication Engineering (ICCCE), 2012 International Conference on*, IEEE, 2012, pp. 712–716.

[5] J. Eberspächer and R. Schollmeier, "First and Second Generation of Peer-to-Peer Systems", in *P2P Systems and Applications*, Springer, 2005.

[6] B. Biskupski, J. Dowling, and J. Sacha, "Properties and mechanisms of self-organizing MANET and P2P systems", *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 2, no. 1, p. 1, 2007.

[7] M. Bisignano, G. Di Modica, *et al.*, "P2P over MANET: A comparison of cross-layer approaches", in *Database and Expert Systems Applications, 2007. 18th International Workshop on*, IEEE, 2007.

[8] *BitTorrent.* [Online]. Available: `http://www.bittorrent.org/beps/bep_0003.html`.

[9] M. K. Sbai, C. Barakat, J. Choi, A. Al Hamra, and T. Turletti, "Adapting BitTorrent to wireless ad hoc networks", in *Ad-hoc, Mobile and Wireless Networks*, Springer, 2008, pp. 189–203.

[10] R. C. Carrano, L. C. S. Magalhães, D. C. M. Saade, and C. V. N. Albuquerque, "IEEE 802.11s multihop MAC: A tutorial", *IEEE Communications Surveys & Tutorials*, vol. 13, no. 1, pp. 52–67, 2011.

[11] S. Kim, O. Lee, S. Choi, and S.-J. Lee, "Comparative analysis of link quality metrics and routing protocols for optimal route construction in wireless mesh networks", *Ad Hoc Netw.*, vol. 9, no. 7, Sep. 2011, ISSN: 1570-8705.

[12] C. Perkins, E. Belding-Royer, and S. Das, *Ad hoc On-Demand Distance Vector Routing*, RFC 3561, Internet Engineering Task Force, Jul. 2003. [Online]. Available: `http://www.ietf.org/rfc/rfc3561.txt`.

[13] M. Guesmia, M. Guezouri, and N. Mbarek, "Performance evaluation of the HWMP proactive tree mode for IEEE 802.11s based wireless mesh networks", in *Communications and Networking (ComNet), 2012 Third International Conference on*, IEEE, 2012, pp. 1–7.

[14] R. G. Garroppo, S. Giordano, and L. Tavanti, "A joint experimental and simulation study of the IEEE 802.11s HWMP protocol and airtime link metric", *International Journal of Communication Systems*, vol. 25, no. 2, pp. 92–110, 2012.

[15] K. S. Nagegowda, H. R. Ranganath, C Puttamadappa, and T. G. Basavaraju, "Performance evaluation of scalable routing protocols using routing metrics for wireless mesh networks under different network scenarios", *IRACST - International Journal of Computer Networks and Wireless Communications (IJCNWC)*, vol. 4, no. 6, 2014.

[16] S. Islam, M. M. Alam, M. A. Hamid, and C. S. Hong, "High throughput path selection for IEEE 802.11s based wireless mesh networks", in *Proceedings of the 4th International Conference on Uniquitous Information Management and Communication*, ACM, 2010.

[17] *Open80211s*. [Online]. Available: `http://open80211s.org/open80211s/`.

[18] *Linux Wireless*. [Online]. Available: `http://wireless.kernel.org/`.

[19] *Minstrel RCA*. [Online]. Available: `http://wireless.kernel.org/en/developers/Documentation/mac80211/RateControl/minstrel`.

[20] R. G. Garroppo, S. Giordano, and L. Tavanti, "Experimental evaluation of two open source solutions for wireless mesh routing at layer two", in *Proceedings of the 5th IEEE international conference on Wireless pervasive computing*, ser. ISWPC'10, Mondena, Italy: IEEE Press, 2010, ISBN: 978-1-4244-6855-3.

[21] Sandvine Intelligent Broadband Networks, "Global Internet Phenomena Report 1H 2014", Tech. Rep., 2014.

[22] B. Cohen, "Incentives build robustness in BitTorrent", First Workshop on the Economics of Peer-to-Peer Systems, 2003.

[23] F. P. D'Elia, G. Di Stasi, S. Avallone, and R. Canonico, "BitTorrent traffic optimization in wireless mesh networks with ALTO service", in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2011 IEEE International Symposium on a*, IEEE, 2011, pp. 1–6.

[24] A. Krifa, M. K. Sbai, C. Barakat, and T. Turletti, “BitHoc: A content sharing application for wireless ad hoc networks”, in *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, IEEE, 2009, pp. 1–3.

[25] A. Marques, F Mira da Silva, and R. Rocha, “P2P over mobile ad-hoc networks”, in *Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops' 09. 6th Annual IEEE Communications Society Conference on*, IEEE, 2009, pp. 1–3.

[26] M. K. Sbai, E. Salhi, and C. Barakat, “A membership management protocol for mobile P2P networks”, in *Proceedings of the 6th International Conference on Mobile Technology, Application & Systems*, ACM, 2009.

[27] M. Rethfeldt, P. Danielis, G. Moritz, B. Konieczek, and D. Timmermann, “Design and Development of a Management Solution for Wireless Mesh Networks based on IEEE 802.11s”, in *Integrated Network and Service Management (IM), 14th IFIP/IEEE Symposium on*, IFIP/IEEE, 2015.

[28] *Vuze BitTorrent client.* [Online]. Available: `http://www.vuze.com`.

[29] *BASS.* [Online]. Available: `https://code.google.com/p/bass-plugin/`.

[30] *Raspberry Pi.* [Online]. Available: `http://www.raspberrypi.org/`.

[31] *rt2x00 Project.* [Online]. Available: `http://rt2x00.serialmonkey.com`.

[32] *Linux Wireless - List of Drivers.* [Online]. Available: `http://wireless.kernel.org/en/users/Drivers`.

[33] *Wireshark.* [Online]. Available: `http://www.wireshark.org/`.

Paper F

DiVote: A Distributed Voting Protocol for Mobile Device-to-Device Communication

Peter Danielis, Sylvia T. Kouyoumdjieva, and Gunnar Karlsson

In Proceedings of the International Teletraffic Congress 28 (ITC), pp. 69-77,
`https://doi.org/10.1109/ITC-28.2016.118`*, ISBN:978-0-9883045-1-2,*
September 2016

DiVote: A Distributed Voting Protocol for Mobile Device-to-Device Communication

Peter Danielis, Sylvia T. Kouyoumdjieva, and Gunnar Karlsson
ACCESS Linnaeus Center, School of Electrical Engineering
KTH Royal Institute of Technology, Stockholm, Sweden
Email: {pdanieli, stkou, gk}@kth.se

Abstract

Distributed aggregation algorithms have traditionally been applied to environments with no or rather low rates of node churn. The proliferation of mobile devices in recent years introduces high mobility and node churn to these environments, thus imposing a new dimension on the problem of distributed aggregation in terms of scalability and convergence speed. To address this, we present DiVote, a distributed voting protocol for mobile device-to-device communication. We investigate a particular use case, in which pedestrians equipped with mobile phones roam around in an urban area and participate in a distributed yes/no poll, which has both spatial and temporal relevance to the community. Each node casts a vote and collects votes from other participants in the system whenever in communication range; votes are immediately integrated into a local estimate. The objective of DiVote is to produce a precise mapping of the local estimate to the anticipated global voting result while preserving node privacy. Since mobile devices may have limited resources allocated for mobile sensing activities, DiVote utilizes D-GAP compression. We evaluate the proposed protocol via extensive trace-driven simulations of realistic pedestrian behavior, and demonstrate that it scales well with the number of nodes in the system. Furthermore, in densely populated areas the local estimate of participants does not deviate by more than 3 % from the global result. Finally, in certain scenarios the achievable compression rate of DiVote is at least 19 % for realistic vote distributions.

1 Introduction

Distributed tasks and computations, e.g., to estimate the average or sum of a set of values, are often conducted based on inputs supplied by collaborative users. Such aggregation functions are of high importance in large-scale distributed systems where there is a need to compute global system properties [1].

In this paper, we focus on a specific class of distributed tasks, namely distributed voting in the context of *urban polling*. Potential urban polling applications collect and process information on locally-relevant questions and provide users in a community with answers to them [2]. Such questions can relate to urban planning optimization ("Is the switching behavior of this traffic light fast enough?") or to safety in a given region ("Do you feel safe in this area?").

In general, the information obtained during a poll can be processed either in a centralized or in a decentralized manner. Centralized processing requires nodes to submit their votes to a central entity. However, this approach lacks scalability and poses privacy concerns as users might in general not want their votes to be seen by a central entity [50]. In particular, this may be of high importance in countries where people do not trust authorities. Contrary, decentralized (or distributed) processing requires nodes to compute local estimates of the result based on partial system knowledge. As opposed to conventional distributed processing scenarios where nodes are considered to be static or semi-static [3], in this work we examine scenarios, in which nodes exhibit high mobility. We consider a node to be a pedestrian carrying some device equipped with a wireless communication interface such as a mobile phone. We rely on device-to-device communication for disseminating votes among participants in the poll. Mobile nodes opportunistically exchange data whenever they come in direct communication range [4]. For the purpose of urban polling, this data comprises voting information conveyed by means of broadcast messages and nodes immediately update their local estimate upon reception of a message.

In the context of distributed voting in urban environments, a distributed voting protocol needs to comply with the following requirements: (1) be scalable, (2) have fast convergence and high accuracy, and (3) preserve node privacy. Thus, in this work we present DiVote, a distributed voting protocol for mobile device-to-device communication, which provides all of the above characteristics. The main contributions are:

- We show that DiVote is suitable for operation in dynamic environments with high node mobility. DiVote makes use of the benefits of D-GAP compression for *scalability* of the protocol [5]. Furthermore, DiVote *preserves node privacy* by applying a cryptographic hash function to user identities.
- We perform extensive trace-driven simulations using realistic pedestrian mobility. We show that DiVote scales well with the number of nodes in the system. Furthermore, DiVote demonstrates both *fast convergence* and *high accuracy*, with local estimates deviating at most by 3 % from the global value in dense scenarios.
- DiVote is able to achieve at least *19 % compression rate* for realistic vote distributions, which makes it appropriate for execution on mobile devices with limited storage capabilities or with restrictions on the memory to be used.
- DiVote exhibits *low processing load* at the application layer as it requires only a fraction of the received broadcast messages (34 % in dense scenarios and even less in sparser scenarios) to be processed to achieve accurate local estimates.

The remainder of this paper is organized as follows: In Section 6, we provide an overview of previous work in the field of distributed aggregation, and reason why current solutions are not suitable for operation in mobile environments. Section 2 introduces DiVote, a distributed voting protocol for mobile device-to-device communication. Section 3 outlines the evaluation scenarios, and Section 5 presents results from realistic pedestrian mobility scenarios. Finally, we conclude the study in Section 7, and present directions for future work.

2 Related Work

Distributed voting belongs to the class of distributed aggregation problems. Distributed aggregation in general comprises computations such as sum, average, minimum, or maximum over unreliable networks, in which no central entity is accessible or required. There are two main paradigms to address this problem, namely *gossip-based* and *tree-based* aggregation. Tree-based aggregation protocols have been shown to perform poorly in dynamic environments with high levels of churn [3], therefore for the rest of this section we focus on discussing the applicability of state-of-the-art gossip-based protocols to our scenario.

Gossip-based aggregation protocols that react to environment changes can be broadly classified in *restarted* and *bookkeeping* protocols.

We first discuss *restarted protocols*, many of which are based on the the push-sum algorithm presented in [6]. The basic idea of the algorithm is that nodes periodically exchange stored values with their neighbors and are thus able to compute the sum or average of all values. However, the main assumption in [6] is that values stay unchanged over time. In [7], the authors propose executing the push-sum algorithm in epochs to reflect changes in the network; the protocol is restarted after each epoch. The distributed random grouping algorithm (DRG) is proposed in [8]. In this algorithm, some nodes can periodically become group leaders and then determine group members by exchanging messages in a handshake manner. The establishment of several roles as well as the information exchange by means of a handshake makes this algorithm too slow to react on changes induced by moving nodes in our envisaged scenario. Another gossip-based distribution estimation approach is suggested in [9]. This algorithm exchanges and merges lists consisting of pairs with value and respective counter between nodes. However, duplicates may occur when applying this approach, which distorts the computed estimate. Most of the restarted gossip algorithms show this shortcoming and do therefore not achieve a high accuracy. In [10], the authors tackle the data duplication problem by simultaneously executing multiple instances of the proposed protocol however the solution exhibits low accuracy [11].

Bookkeeping gossip-based protocols are able to revert changes in the nodes' states. In [12], a node saves the states on its neighbors and recovery is triggered when a node crashes or disappears. Here, a tradeoff between accuracy and protocol overhead has to be chosen so either scalability in terms of memory consumption or accuracy are decreased. In

Table 1: Comparison of DiVote with state-of-the-art approaches for dynamic environments.

Feature	Tree-based	Restarted gossip	Bookkeeping gossip	DiVote
Convergence speed	Low	Low-medium	Low	High
Accuracy	High	Low-medium	High	High
Scalability	Low	High	Low-medium	High
Privacy	Depends on extra measures			Yes

[13], the authors propose LiMoSense, an algorithm for live monitoring in dynamic sensor networks, which takes into account node churn and link failures at runtime. LiMoSense is not appropriate for dynamically changing environments, since it assumes a known set of neighbors, from which it randomly chooses a single neighbor at a time for message exchange. In [14], the authors present Flow Updating, a bookkeeping algorithm, which iteratively averages values towards the global network mean. Every node computes a flow value for each of its neighbors and stores the value in a matrix. The algorithm tries to enforce skew symmetry of this matrix. This however significantly decreases the convergence speed. To summarize, bookkeeping protocols usually require up to thousands of rounds to converge, thus they are not applicable in scenarios with high dynamics.

In Section 1, we outlined the main characteristics of a protocol suitable for distributed aggregation in dynamic environments. In Table 1, we compare tree-based, restarted, and bookkeeping gossip-based protocols with respect to these characteristics. We also show how our protocol DiVote compares to others in the literature. In the next section, we present DiVote in detail.

3 DiVote: A Distributed Voting Protocol

In this section, we present DiVote, a distributed voting protocol for mobile device-to-device communication. We begin by introducing some of the building blocks of the protocol, i.e., the vector compression scheme and the fundamental vector operations that are introduced by the protocol. We then present the details of the algorithm behind DiVote.

3.1 The need for D-GAP compression

In the context of distributed voting, each node can cast a binary vote (0 or 1) to a poll. (We note that the assumption of binary votes does not limit the reasoning to follow, and can easily be extended to any number of votes. In this paper, we only consider binary votes for the sake of brevity.) For nodes to be able to calculate the anticipated global vote, they need to keep track of the votes of other peers in their vicinity throughout their lifetime. However, keeping track of a simple moving average may result in votes being counted multiple times if a node and a peer come in communication range more than once. Furthermore, keeping track of votes in the form of a binary vector may be consuming a lot of resources, especially if the vector contains long sequences of 0s or 1s. To address these problems, in DiVote we leverage the concept of *D-GAP compression* [5]. D-GAP compression provides a compressed representation of bit vectors in the form of integer vectors (later referred to as D-GAP vectors) and can be treated as a specialized variant of run length encoding. Each integer in a D-GAP vector represents the number of consecutive 0s or 1s that are present in the bit vector at a given position. Whether the integer corresponds to a sequence of 0s or 1s is determined by the leading bit of the D-GAP vector. A leading bit of 0 shows that the first integer corresponds to a number of consecutive 0s, followed by a number of consecutive 1s and so on; a leading bit of 1 indicates the opposite behavior.

An example of converting a 12-bit vector into a 9-bit D-GAP vector is illustrated in Figure 1. We calculate the total number of bits required for representing the D-GAP vector, N(DGAP), as follows:

$$\mathrm{N(DGAP)} = \sum_{i=1}^{n} (\lfloor \log_2 d_i \rfloor + 1) \tag{1}$$

where d_i is the integer representation at position i of the D-GAP vector, and n is the size of the vector.

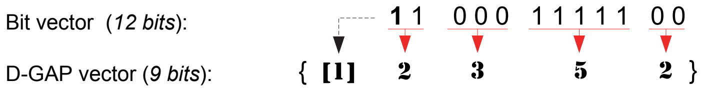


Figure 1: An example of D-GAP compression. A bit vector of 12 bits is converted into an integer D-GAP vector of 9 bits. The leading bit in the D-GAP vector indicates if the vector starts with 0s or 1s.

For decoding purposes, each D-GAP vector needs to have a corresponding D-GAP *mask* vector. In essence, a D-GAP mask vector is a bit vector of consecutive sequences of 0s and 1s, and each sequence indicates the boundaries of an integer in the D-GAP vector. Let us assume the following D-GAP vector: {[0] 2 1} as an illustration of the problem. For the vector to be stored in memory it will be converted to {[0] 10 1}. However this representation alone is not enough for decoding, i.e., it is impossible to tell whether the original D-GAP vector was {[0] 2 1} or {[0] 5}. To decode the D-GAP vector, we need to apply a mask vector {[0] 001}. The mask vector shows that the initial two bits correspond to the first integer while the third bit corresponds to the second integer. For longer D-GAP vectors, the D-GAP vector mask will iterate between sequences of consecutive 0s and sequences of consecutive 1s.

3.2 Operations on D-GAP vectors

A D-GAP vector is solely a data structure. Hence, we define the following three operations for DiVote that can be performed on two D-GAP vectors of arbitrary lengths: `merge`, `consolidate`, and `append`. For brevity, let us consider vectors of different lengths, DGAP_{min} and DGAP_{max}, denoting the shorter and the longer vector, respectively. Note that here vector length corresponds to the *expanded bit vector length* and is defined as $\mathrm{N(BVEC)} = \mathrm{L(DGAP)} = \sum_{i=1}^{n} d_i$.

- The `merge` operation combines DGAP_{min} and $\mathrm{DGAP}_{max}[1{:}\mathrm{L}(\mathrm{DGAP}_{min})]$ vectors into a resulting vector DGAP_{res}. The `merge` operation is performed in an iterative manner until it reaches the end of DGAP_{min}.
- The `consolidate` operation combines the last position of DGAP_{res} with the next position of DGAP_{max} after the `merge` operation is performed. The `consolidate` operation is only performed if the integers at these two positions correspond to the same bit value.
- The `append` operation simply adds the remainder of DGAP_{max} to DGAP_{res}.

Observe that if the two input vectors are of equal lengths, the only operation that will be performed is `merge`. Figure 2 illustrates an example of all three operations that can be performed with D-GAP vectors.

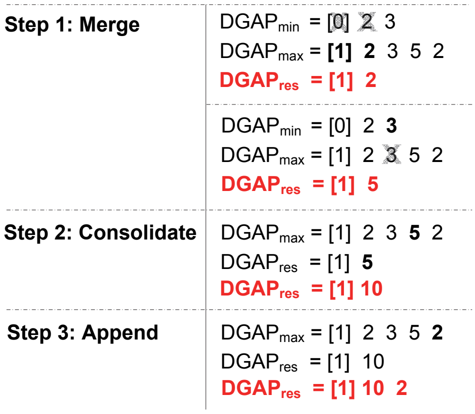


Figure 2: Merging, consolidating, and appending D-GAP vectors. Bold integers denote positions under consideration, crossed integers are not considered, and the resulting vector is highlighted in red.

3.3 Functional principles of DiVote

With DiVote, each node locally keeps track of nodes it has obtained knowledge of, either directly or via other peers, as well as of their votes. This information is presented in the form of two correlated D-GAP vectors. Whenever a node first casts a vote, it adds itself

```
{
  "message" : {
    "node-id" : "0x00:0a:95:9d:68:16",
    "divote" : [ {
      "poll" : {
        "poll-id" : "5",
        "topic" : "sample poll",
        "shared" : [0 24 1],
        "votes" : [0 25],
        "updated" : "2016-02-17T18:30:02Z"
      }
    } ]
  }
}
```

Listing 1: An example of a DiVote message in the JSON format.

to the *shared-nodes vector*, and it adds its vote to the *votes vector*. Observe that due to privacy preservation reasons, the position, in which information is stored in each of the D-GAP vectors, is determined by a cryptographic hash function such as MD5, which is calculated over a unique identifier, e.g., the node's MAC address. For instance, if for a node, which casts a vote for 1, the cryptographic hash function returns a position of 25, both its shared-nodes vector and its votes vector would be initialized with $\{[0]\ 24\ 1\}$. If the same node were to cast a vote for 0, the votes vector would instead be initialized with $\{[0]\ 25\}$. If a node is compromised, it could potentially associate the vote to the respective node during such initialization. However, in this work we assume all nodes to be trustworthy.

Note that hash functions can compute the same hash value for different identifiers, i.e., collisions can occur. This would lead to positions that some nodes would share, i.e., some nodes would replace their information when shared alternately, which could falsify their local estimate. However, the collision probability of MD5, which computes 128-bits hash values, is as low as $2.7 \cdot 10^{-20}$ when, e.g., calculating hash values from 2^{32} values (assuming the birthday paradox [15]) and is therefore neglected.

Each node periodically broadcasts a beacon containing its shared-nodes vector and its votes vector to peers in its vicinity. An example of a DiVote beacon message is presented in Listing 1. Whenever a node receives information from another peer, it immediately updates both its shared-nodes vector and its votes vector following the DiVote protocol outlined in Algorithm 1. We note that the procedure in Algorithm 1 is performed simultaneously with respect to both vectors. However, here we only show how new votes are incorporated into the votes vector for the sake of brevity. Whenever a node receives a beacon from another peer in proximity, it first extracts the received information and checks in a local database whether the received vector from peer i, $\mathrm{DGAP}^{(i)}_{rec}$, has changed (line 4). This check is currently done by looking up the `node-id` in the local database first. If the `node-id` is found, there has been prior communication with this peer and

Algorithm 1 The DiVote Protocol

```
1: $\text{DGAP}^{(i)}_{rcv} \leftarrow$ received DGAP vector from node $i$
2: $\text{DGAP}_{loc} \leftarrow$ local DGAP vector
3: $\text{DGAP}_{res} \leftarrow$ resulting DGAP vector
4: if $\text{DGAP}^{(i)}_{rcv}$ changed since last beacon from node $i$ then
5:     $\text{DGAP}_{min} = \min(\text{DGAP}_{loc}, \text{DGAP}^{(i)}_{rcv})$
6:     $\text{DGAP}_{max} = \max(\text{DGAP}_{loc}, \text{DGAP}^{(i)}_{rcv})$
7:     while ! $\text{DGAP}_{min}.end()$ do
8:         $\text{DGAP}_{res} = \textsc{Merge}(\text{DGAP}_{min}, \text{DGAP}_{max})$
9:         $rpos \leftarrow$ end position in $\text{DGAP}_{res}$
10:        $mpos \leftarrow$ current position in $\text{DGAP}_{max}$
11:    end while
12:    $\text{state}_{res} = (rpos \mod 2)\ xor\ \text{DGAP}_{res}[0]$
13:    $\text{state}_{max} = (mpos \mod 2)\ xor\ \text{DGAP}_{max}[0]$
14:    if $\text{state}_{res} == \text{state}_{max}$ then
15:        $\text{DGAP}_{res} = \textsc{Consolidate}(\text{DGAP}_{res}[rpos], \text{DGAP}_{max}[mpos])$
16:    end if
17:    $\text{DGAP}_{res} = \textsc{Append}(\text{DGAP}_{res}, \text{DGAP}_{max}[mpos : \text{end}])$
18:    $\text{DGAP}_{loc} \leftarrow \text{DGAP}_{res}$
19: end if
```

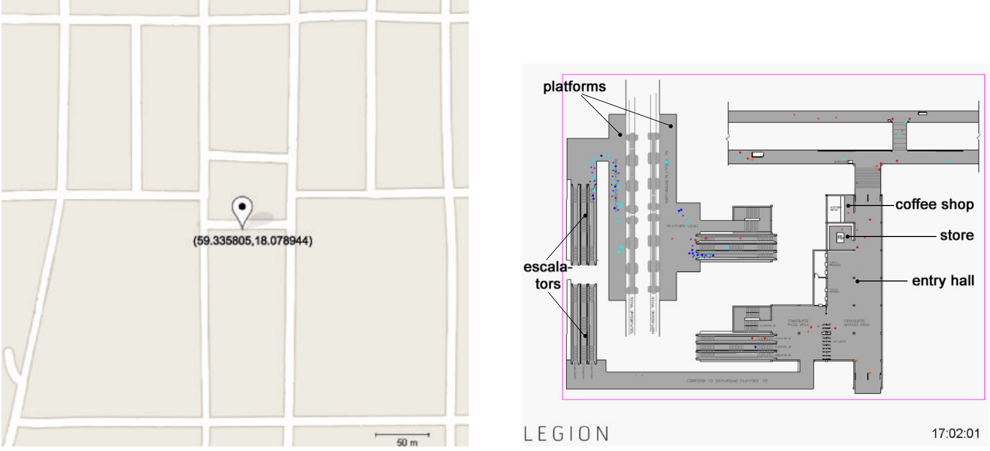
the advertised `updated` field value is compared to the previously registered `updated` field value. If they do not differ, the local vector DGAP_{loc} stays unchanged. Otherwise, DiVote consecutively executes the operations `merge` (lines 7-11), `consolidate` (lines 12-16) and `append` (line 17) in order to update the local estimate (line 18). Thus, the shared-nodes vector and the votes vector contain cumulative information of all nodes that have been shared over time, and their corresponding votes, even if these nodes have left the system. Furthermore, DiVote allows nodes to disclose peers that they have not encountered physically by propagating the knowledge accumulated by other participants in the system. This allows DiVote to achieve fast convergence and high accuracy in a distributed manner in scenarios with high levels of mobility, such as in urban environments. Finally, correlating votes and nodes is impossible by third party entities such as central collection points or compromised participants in the poll.

4 Evaluation Scenario

In this section, we introduce the mobility scenario as well as the simulation setup and investigated performance metrics.

4.1 Mobility scenario

In order to realistically recreate pedestrian mobility, we use the Walkers traces [16] captured in Legion Studio [17], a commercial simulator initially developed for designing and dimensioning large-scale spaces via simulation of pedestrian behaviors. Its multi-agent pedestrian model is based on advanced analytical and empirical models which have been calibrated by measurement studies. Each simulation run results in a trace file, containing a snapshot of the positions of all nodes in the system every 0.6 s.



(a) (b)

Figure 3: Urban scenarios: (a) a grid of streets representing a part of downtown Stockholm, Östermalm, and (b) a two-level subway station.

Fig. 4 (b) and 4 (c) present the scenarios considered in our evaluation: an outdoor urban scenario, modeling the Östermalm area of central Stockholm, and an indoor scenario, recreating a two-level subway station. We note that it is not possible to capture all states of human mobility with a single setup, however the scenarios are representative of typical daytime pedestrian mobility.

The Östermalm scenario consists of a grid of interconnected streets. Fourteen passages connect the observed area to the outside world. The active area, i.e., the total surface of the streets, is 5872 m^2. The nodes are constantly moving, hence the scenario can be characterized as a high mobility scenario.

The Subway station has train platforms connected via escalators to the entry-level. Nodes arrive on foot from any of five entries, or when a train arrives at the platform. The train arrivals create burstiness in the node arrivals and departures. Nodes congregate while waiting for a train at one of the platforms, or while taking a break in the store or the coffee shop at the entry level. The active area is 1921 m^2.

If not stated otherwise, the input parameters of the Östermalm and the Subway scenario result in approximately the same mean node density of 0.1 nodes/m^2. (More information can be found in [18].)

4.2 Simulation setup

In our evaluation scenarios, we assume that all nodes carry devices and all are participating in the distributed poll in the area. Each node casts a binary vote $v = \{0, 1\}$ upon entry in the simulation, and votes are distributed according to a distribution $f(x)$ with a mean $\mathbb{E}(x)$.

For the evaluation, we use an implementation of an opportunistic content distribution system in the OMNeT++ simulator [19]. Each simulation run is executed in synchronous rounds of 0.6 s which corresponds to the granularity of the mobility traces we use. Nodes broadcast their shared-nodes vector and their votes vector at the beginning of each round. To avoid collisions on the wireless medium, the broadcast transmission of each node in each round is distributed uniformly at random $\mathcal{U}(0, 0.5)$ s. The transmission range is set to 10 m.

4.3 Performance metrics

We focus on evaluating the following performance metrics.

- *Deviation* Δ: The deviation is a measure of the accuracy of the DiVote protocol, i.e., it shows how close the local estimate of a node is to the anticipated global result. The deviation is calculated as:

  $$\Delta = \left| \frac{\bar{x} - x}{x} \right| \tag{2}$$

  where $\bar{x} = \mathbb{E}(\mathrm{DGAP}_{loc})$ is the local estimate, and x is the anticipated global result depending on the nodes currently in the system.

- *Compression ratio* (CR): The compression ratio is a measure of the efficiency and scalability of the DiVote protocol in terms of resource management, i.e., how much storage space does the protocol require for performing distributed voting computations in a mobile environment. The compression ratio is calculated as:

  $$\mathrm{CR} = 1 - \frac{\mathrm{N(DGAP)}}{\mathrm{N(BVEC)}} \tag{3}$$

  where N(DGAP) is calculated as per Eq. 1, and N(BVEC) is the number of bits required if the data were represented in the form of a bit vector.

- *Information overhead* (IO): The information overhead is a measure of the processing load reduction for a node in the system and therefore indicates scalability as well. It shows how many of the received broadcasts do not need to be processed. The information overhead is calculated at the application layer as:

  $$\mathrm{IO} = 1 - \frac{\mathrm{n(BRC)}}{\mathrm{N(BRC)}} \tag{4}$$

  where N(BRC) is the total number of broadcast messages received by a node throughout its lifetime in the system, and $\mathrm{n(BRC)} \subset \mathrm{N(BRC)}$ is the number of broadcast messages that were used for updating the local estimate of the node.

5 Simulation Results

In this section, we investigate simulation results for:

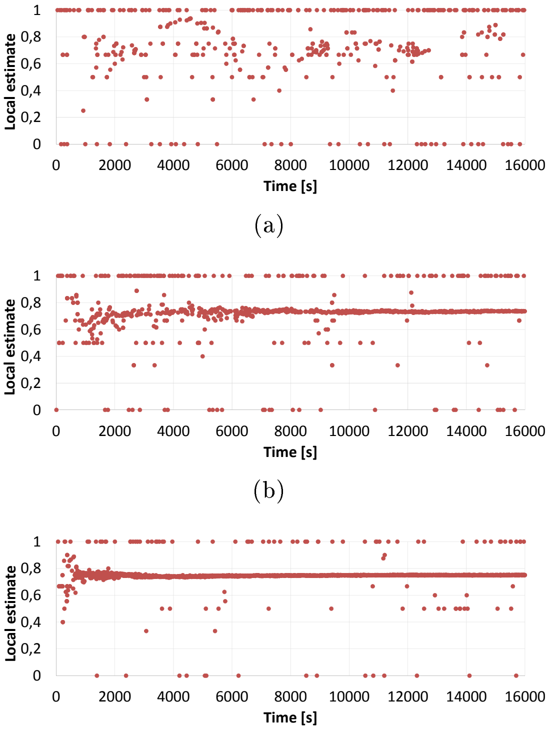
- different arrival rates λ while fixing the scenario and the distribution of nodes voting for one;
- two different scenarios while fixing the arrival rate λ and the distribution of nodes voting for one;
- two different distributions of nodes voting for one while fixing the arrival rate λ.

5.1 Effect of arrival rate

First, we present results for the Östermalm scenario for the arrival rates $\lambda = \{0.0025, 0.005\ 0.01, 0.07, 0.15, 0.30\}$ nodes/s. We assume that votes are deterministically distributed, with a mean $\mathbb{E}(x) = 0.75$, i.e., 75 % of all nodes vote for one and 25 % vote for zero. (We release this assumption in Section 5.3.) In this case, the first node entering the system votes for zero whereas the following three nodes vote for one. After that, this distribution continues for all further nodes. As we will see in Section 5.2, this represents the worst case of achievable compression rate.

Figures 4 (a)-(c) show the local estimates of all nodes over time for sparsely populated scenarios, i.e., $\lambda = \{0.0025, 0.005, 0.01\}$ nodes/s. We see that the convergence of the local estimates towards the global result is strongly dependent on the population density. For low values of λ, Figure 4 (a), nodes are not able to estimate correctly the expected global result. As the arrival rate increases, Figure 4 (b), a clearer trend can be seen towards convergence, and at $\lambda = 0.01$ nodes/s nodes are able to locally estimate the global result. Still, for any of the low arrival rates, some nodes do not gain sufficient knowledge about other nodes or even do not disclose anyone so that their local estimates remain 0 or 1 (see Figures 4 (a)-(c)). As the arrival rate further increases, nodes converge earlier to the global result, and outliers disappear. Thus, we omit results for $\lambda = \{0.07, 0.15, 0.3\}$ nodes/s for the sake of brevity.

Figures 5 (a)-(c) illustrate the proportion of shared nodes over time. In sparser scenarios, Figure 5 (a), the proportion of shared nodes is kept below 20 %. Furthermore, information on shared nodes is continuously lost when nodes leave the system. Therefore, we are not able to observe clear convergence of the local estimates towards the global results, Figure 4 (a). With increase of the arrival rate, $\lambda = 0.005$ nodes/s, up to 60 % of all nodes in the system are shared leading to a clearer convergence towards the global result, Figure 5 (b). At around t = 7000 s when approximately 30 % of all nodes have already been shared, the trend towards the mean $\mathbb{E}(x) = 0.75$ becomes apparent. We note that approximately 60 % of nodes are already shared around t = 4000 s by some of the nodes in the system. A further investigation shows that indeed the knowledge is kept only in two nodes which have shared around 130 nodes each. However, both of these nodes leave the system in t $\in$ (4600, 4700) s so their accumulated knowledge is lost and the process is restarted. This is reflected in Figure 4 (b) as well as the trend towards the global result

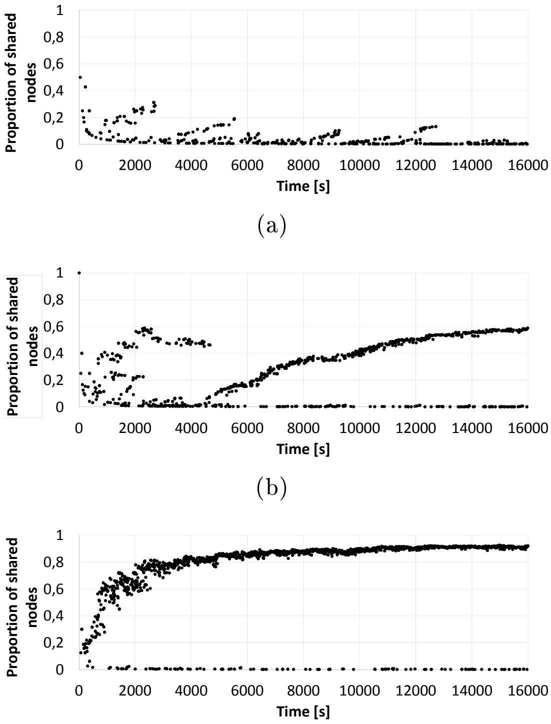


(a)

(b)

(c)

Figure 4: Local estimates of all nodes for the Östermalm scenario: (a) $\lambda = 0.0025$ nodes/s, (b) $\lambda = 0.005$ nodes/s, and (c) $\lambda = 0.01$ nodes/s.



(a)

(b)

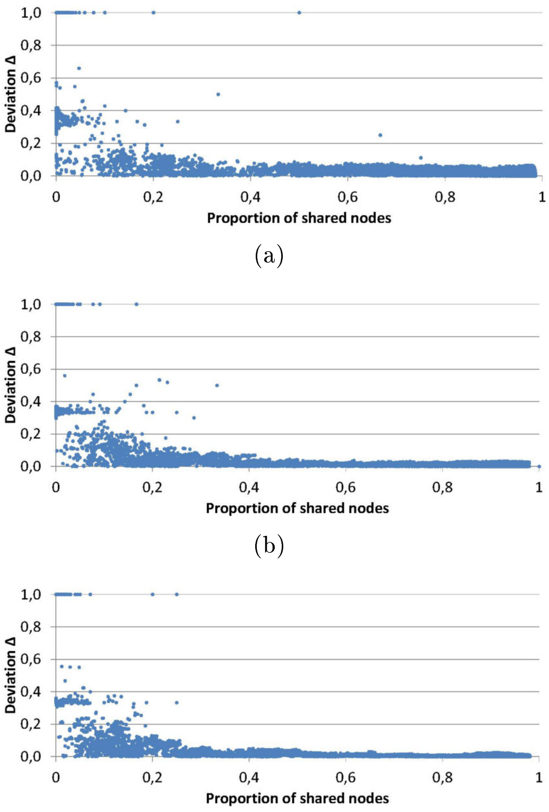
(c)

Figure 5: Proportion of shared nodes for the Östermalm scenario: (a) $\lambda = 0.0025$ nodes/s, (b) $\lambda = 0.005$ nodes/s, and (c) $\lambda = 0.01$ nodes/s.

does not become clear before t = 7000 s when the vast majority of nodes has shared at least 30 % of all nodes. Finally, Figure 5 (c) shows an even clearer convergence trend; already at t = 1000 s when approximately 30 % of all nodes have been shared most nodes approach the mean $\mathbb{E}(x) = 0.75$. Thus, we conclude that even in dynamically changing environments there is a correlation between the percentage of shared nodes and the convergence to the global result.

The observation that disclosing approximately 30 % of nodes is sufficient for achieving precise estimate of the global result is further confirmed in Figures 6 (a)-(c), which show the change in deviation Δ with respect to the proportion of shared nodes for the denser Östermalm scenarios with $\lambda = \{0.07, 0.15, 0.3\}$ nodes/s. As the arrival rate increases, the deviation Δ significantly decreases once 30 % of the nodes have been shared, from 15 % for $\lambda = 0.07$ nodes/s, Figure 6 (a), to below 6 % for $\lambda = 0.3$ nodes/s, Figure 6 (c).

We further evaluate the performance of the system in steady state, i.e., once the average number of nodes in the area stays unchanged despite the arrivals and departures in the system. We then aggregate results from a 1000 nodes. We subsequently exclude the sparse scenario with $\lambda = 0.0025$ nodes/s from consideration as the trace does not comprise 1000 nodes after the steady state has been reached, and local estimates do not converge to the global result.



(a)

(b)

(c)

Figure 6: Deviation Δ depending on the proportion of shared nodes for the Östermalm scenario: (a) $\lambda = 0.07$ nodes/s, (b) $\lambda = 0.15$ nodes/s, and (c) $\lambda = 0.3$ nodes/s.

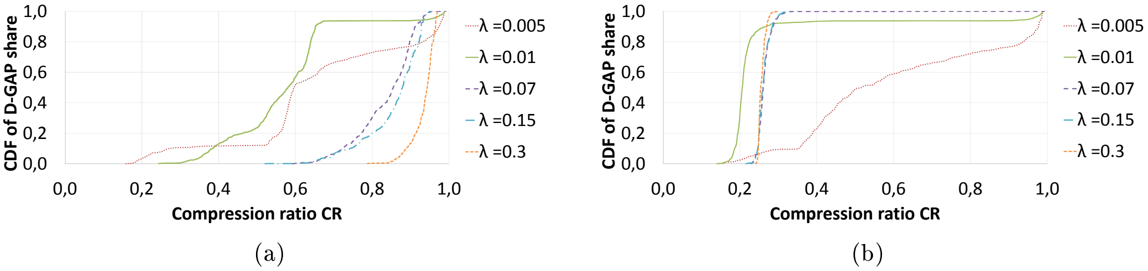
Table 2: Average and maximum deviation Δ, and information overhead (IO) for the Östermalm scenario with different arrival rates λ.

Arrival rate λ [nodes/s]	Avg. Δ [%]	Max. Δ [%]	IO [%]
0.005	14.91±1.19	100	93.30±0.4
0.01	7.19±0.73	100	94.05±0.2
0.07	2.06±0.08	6.3	93.98±0.3
0.15	1.01±0.05	3.07	89.55±0.2
0.3	0.79±0.04	2.43	66.28±0.5

Table 3: Comparison: Average and maximum deviation Δ and information overhead (IO) for the Östermalm with $\lambda = 0.15$ nodes/s and Subway scenarios.

Scenario	Avg. Δ [%]	Max. Δ [%]	IO [%]
Östermalm	1.01±0.05	3.07	89.55±0.2
Subway	0.79±0.04	3.27	76.80±0.8

Table 3 shows the average and maximum deviation Δ including 95 % confidence intervals as well as the information overhead in the steady state depending on the arrival rate λ. Note that minimum deviation values are omitted as they are zero in all cases. These results clearly show that the denser the scenario, the smaller the deviation Δ, i.e., the accuracy increases. Note that for $\lambda = 0.005$ nodes/s and $\lambda = 0.01$ nodes/s some nodes are still completely wrong regarding their local estimate when they leave the system. As the arrival rate increases, both the average and the maximum deviation are steadily decreasing. Finally, for the most densely populated scenario, $\lambda = 0.3$ nodes/s, the maximum deviation never exceeds 3 %. Moreover, the information overhead is between 93 % and 94 % for $\lambda = \{0.005, 0.01, 0.07\}$ nodes/s, which results from the fact that often the same nodes meet again and do not exchange new information. On the one hand, this underlines the low processing load on application layer and thus DiVote's scalability in sparse scenarios. On the other hand, as shown above, the accuracy is not very high for the sparse scenarios. In denser scenarios, the information overhead amounts to lower values of 90 % and 66 % for $\lambda = 0.15$ nodes/s and $\lambda = 0.3$ nodes/s, respectively. Due to the higher population density, it is more probable that new nodes meet, which then exchange new information and thus the information overhead decreases.



(a) (b)

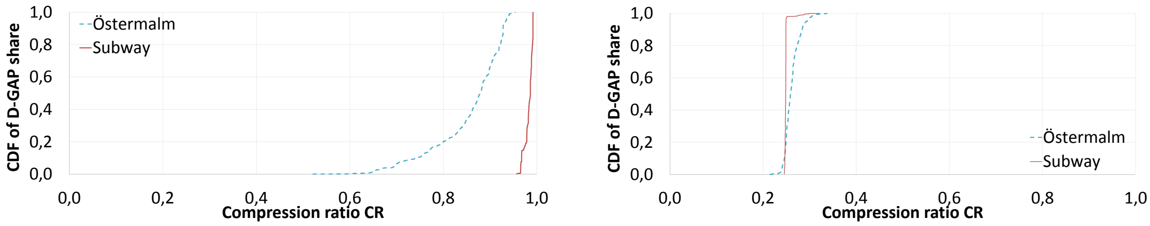
Figure 7: CDF of the compression ratio for storing (a) shared nodes and (b) votes in the Östermalm scenario under different arrival rates λ.

Figure 7 shows the cumulative distribution functions (CDFs) of the compression ratio when storing the D-GAP vectors with shared nodes and with votes across different arrival rates λ. The achievable compression ratio is higher in case of using D-GAP for storing shared nodes, Figure 7 (a), than that for storing votes, Figure 7 (b). For each shared node, a 1 is set in the D-GAP, which results in long sequences of consecutive 1s and increases compression. On the other hand, depending on the voting distribution and its mean value, sequences of consecutive 1s may be shorter in the D-GAP vector for storing votes resulting in a lower compression ratio. In Figure 7 (a), the curves for $\lambda = 0.005$ nodes/s and $\lambda = 0.01$ nodes/s show an erratic trend as the scenarios are too sparsely populated to show clear convergence. With the increase of the arrival rate, however, the compression ratio also increases, and for $\lambda = 0.3$ nodes/s, the average compression ratio amounts to 92 %. This results from the fact that most nodes in the system have been shared, thus almost all bits in the D-GAP are set to 1. The compression ratio of the D-GAP for storing votes approaches 25 % as the arrival rate increases, Figure 7 (b). This behavior is strongly dependent on the chosen voting distribution as well as on its mean value $\mathbb{E}(x)$

= 0.75. As mentioned earlier, the first node entering the system always votes for zero, while the following three nodes vote for one, and subsequently the distribution applies to all further nodes. Hence, the D-GAP for storing votes convergences to sequences of three consecutive 1s, which are interrupted by a 0. Applying Equation 1, three consecutive 1s and one 0 requires 3 bits for storage whereas a plain bit vector would require 4 bits. Therefore, the achievable compression ratio convergences to a value of 0.25 while more and more nodes are shared, which represents the worst case in terms of compression ratio. As we show in Section 5.3, when the voting distribution follows a different distribution, the achievable compression ratio increases.

5.2 Effect of scenario

We now investigate the impact of different topologies by comparing the performance of the Östermalm and the Subway scenario. Table 3 shows the average and maximum deviation Δ including their confidence intervals, as well as the information overhead in steady state. Again, the minimum deviation values are omitted as they are zero in all cases. These results clearly show that both scenarios exhibit high accuracy, namely achieving an average deviation of 1 % for the Östermalm and the Subway scenario. The lower deviation in the Subway scenario is due to the bursty arrivals in the system. This also results in a lower information overhead of approximately 77 % compared to 90 % in case of the Östermalm scenario as node departures are also bursty. As the sojourn time of nodes is lower in the Subway than in the Östermalm scenario, nodes exchange less messages but those exchanged are useful for advancing the knowledge of other nodes.



(a) (b)

Figure 8: Comparison: CDF of the compression ratio for storing (a) shared nodes and (b) votes in the Östermalm scenario with for $\lambda = 0.15$ nodes/s and the Subway scenario.

Figure 8 shows the CDF of the compression ratio for storing shared nodes and votes in both scenarios. The Subway scenario exhibits higher compression ratio as nodes are quicker to disclose all other nodes in the system due to the smaller and more confined area, in which mobility occurs, Figure 8 (a). Due to the same fact, the compression ratio of the D-GAPs for storing votes more closely approaches a value of 0.25 in the Subway scenario as apparent from Figure 8 (b).

5.3 Effect of voting distribution

Finally, we compare the performance of DiVote for different voting distributions. We consider a deterministic as well as a uniform voting distribution, which can be seen as a more realistic representation of votes of pedestrians in an area. We choose three different mean values of the distribution ($\mathbb{E}(x) = 0.25$, 0.5, and 0.75), and we perform five simulation runs for each mean value.

Table 4 shows the average compression ratio with 95 % confidence intervals for $\mathbb{E}(x) =$ 0.25, 0.5, and 0.75 in the Östermalm scenario. These results obtained for the uniform distribution are contrasted with the deterministic distribution. As the mobility trace is unchanged during simulation runs, the compression ratio for the D-GAP vector for shared nodes is independent of the proportion of nodes that vote for one. However, uniformly distributing the votes leads to higher compression ratios compared to the deterministic distribution as longer sequences of 1s and 0s are possible.

Table 4: Comparison of D-GAP compression ratios (CRs) for the Östermalm scenario with $\lambda = 0.15$ nodes/s. Uniform and deterministic distribution are contrasted with each other.

	Uniform distribution		Deterministic distribution	
$\mathbb{E}(x)$	Avg. CR of D-GAPs		Avg. CR of D-GAPs	
	Shared nodes	Votes	Shared nodes	Votes
0.25	0.86±0.0003	0.38±0.019	0.86±0.004	0.32±0.002
0.5		0.23±0.005		0.08±0.002
0.75		0.34±0.012		0.26±0.001

The gain in terms of achievable compression ratio becomes more apparent from Table 5, which shows the average compression ratio and their 95 % confidence intervals for the Subway scenario. As expected, in case of the deterministic distribution the compression ratio approaches a value of 0 % for $\mathbb{E}(x) = 0.5$ as the D-GAP expands to a bit vector of alternating 0 and 1. However, when votes are uniformly distributed, longer sequences of 1s and 0s become possible resulting in a compression ratio of 19 % even for an average global result of $\mathbb{E}(x) = 0.5$.

6 Conclusion

In this paper, we presented DiVote, a distributed voting protocol in the context of urban polling, which is suitable for environments, in which nodes exhibit high mobility. DiVote relies on device-to-device communication to exchange voting information. The proposed DiVote protocol exhibits the following main features:

- *Privacy*: By using a cryptographic hash function, votes cannot be related to the corresponding node identities.

Table 5: Comparison of D-GAP compression ratios (CRs) for the Subway scenario. Uniform and deterministic distribution are contrasted with each other.

	Uniform distribution		Deterministic distribution	
$\mathbb{E}(x)$	Avg. CR of D-GAPs		Avg. CR of D-GAPs	
	Shared nodes	Votes	Shared nodes	Votes
0.25	0.98±0.00001	0.33±0.01	0.98±0.0005	0.25±0.0004
0.5		0.19±0.01		0.001±0.0005
0.75		0.33±0.01		0.25±0.0003

- *Convergence speed*: The dynamism due to mobility imposes tight constraints on the convergence speed of the algorithm. Consequently, DiVote immediately updates the local estimate. Simulation results obtained when applying DiVote to realistic pedestrian mobility traces show that even in sparse scenarios local estimates quickly converge to the global result after having shared 30 % of all nodes in the system.
- *Accuracy*: At the same time, accuracy of local estimates is ensured as DiVote avoids to erroneously count votes multiple times, which is of decisive importance as the same node may be encountered several times. In dense scenarios, the local estimate does not deviate by more than 3 % from the global result after the system reaches the steady state.
- *Scalability*: Rather than storing shared nodes and their votes in plain bit vectors, DiVote uses D-GAP compression to be scalable in terms of required storage capacity. For realistic voting distributions, at least 19 % compression is achieved. Furthermore, the processing overhead introduced at the application layer is very low as only a fraction of the received messages (approximately 30 %) has to be processed even in the densely populated scenarios.

Prospectively, we intend to theoretically analyze DiVote and compare it with other existing schemes. We will further investigate more voting distributions and compare the D-GAP compression with other compression algorithms.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) (research fellowship, GZ: DA 1687/2-1) for their financial support.

7 References

[50] Y. Benkaouz, R. Guerraoui, M. Erradi, and F. Huc, “A distributed polling with probabilistic privacy”, in *Reliable Distributed Systems (SRDS), 2013 IEEE 32nd International Symposium on*, 2013, pp. 41–50.

[1] S. Gambs, R. Guerraoui, H. Harkous, F. Huc, and A.-M. Kermarrec, “Scalable and secure polling in dynamic distributed networks”, in *Reliable Distributed Systems (SRDS), 2012 IEEE 31st Symposium on*, 2012, pp. 181–190.

[2] L. Koeman, V. Kalnikaitė, and Y. Rogers, “"everyone is talking about it!": A distributed approach to urban voting technology and visualisations”, in *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, ser. CHI ’15, Seoul, Republic of Korea: ACM, 2015, pp. 3127–3136, ISBN: 978-1-4503-3145-6.

[3] L. Nyers and M. Jelasity, “A comparative study of spanning tree and gossip protocols for aggregation”, *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, pp. 4091–4106, 2015, cpe.3549, ISSN: 1532-0634.

[4] Ó. Helgason, E. A. Yavuz, S. Kouyoumdjieva, L. Pajevic, and G. Karlsson, “A mobile peer-to-peer system for opportunistic content-centric networking”, in *Proc. ACM SIGCOMM MobiHeld workshop*, New Delhi, India, 2010.

[5] A. Kuznetsov, *D-gap compression*, 2002. [Online]. Available: `http://bmagic.sourceforge.net/dGap.html`.

[6] D. Kempe, A. Dobra, and J. Gehrke, “Gossip-based computation of aggregate information”, in *Foundations of Computer Science, 2003. Proceedings. 44th Annual IEEE Symposium on*, 2003, pp. 482–491. DOI: `10.1109/SFCS.2003.1238221`.

[7] M. Jelasity, A. Montresor, and O. Babaoglu, “Gossip-based aggregation in large dynamic networks”, *ACM Trans. Comput. Syst.*, vol. 23, no. 3, pp. 219–252, Aug. 2005, ISSN: 0734-2071.

[8] J.-Y. Chen, G. Pandurangan, and D. Xu, “Robust computation of aggregates in wireless sensor networks: Distributed randomized algorithms and analysis”, *Parallel and Distributed Systems, IEEE Transactions on*, vol. 17, no. 9, pp. 987–1000, 2006, ISSN: 1045-9219.

[9] M. Haridasan and R. van Renesse, “Gossip-based distribution estimation in peer-to-peer networks”, in *Proceedings of the 7th International Conference on Peer-to-peer Systems*, ser. IPTPS’08, Tampa Bay, Florida: USENIX Association, 2008, pp. 13–13.

[10] J. Sacha, J. Napper, C. Stratan, and G. Pierre, “Adam2: Reliable distribution estimation in decentralised environments”, in *Distributed Computing Systems (ICDCS), 2010 IEEE 30th International Conference on*, 2010, pp. 697–707.

[11] M. Borges, P. Jesus, C. Baquero, and P. S. Almeida, “Spectra: Robust estimation of distribution functions in networks”, *CoRR*, vol. abs/1204.1373, 2012.

[12] F. Wuhib, M. Dam, R. Stadler, and A. Clem, “Robust monitoring of network-wide aggregates through gossiping”, *Network and Service Management, IEEE Transactions on*, vol. 6, no. 2, pp. 95–109, 2009, ISSN: 1932-4537.

[13] I. Eyal, I. Keidar, and R. Rom, “Limosense: Live monitoring in dynamic sensor networks”, *Distrib. Comput.*, vol. 27, no. 5, pp. 313–328, Oct. 2014, ISSN: 0178-2770. DOI: `10.1007/s00446-014-0213-8`.

[14] P. Jesus, C. Baquero, and P. S. Almeida, “Flow updating: Fault-tolerant aggregation for dynamic networks”, *Journal of Parallel and Distributed Computing*, vol. 78, pp. 53 –64, 2015, ISSN: 0743-7315. DOI: `http://dx.doi.org/10.1016/j.jpdc.2015.02.003`.

[15] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction To Algorithms*. MIT Press, 2001.

[16] S. T. Kouyoumdjieva, Ó. R. Helgason, and G. Karlsson, *CRAWDAD data set kth/walkers (v. 2014-05-05)*, Downloaded from http://crawdad.org/kth/walkers/, May 2014.

[17] *Legion Studio*, `http://www.legion.com/`.

[18] Ó. Helgason, S. T. Kouyoumdjieva, and G. Karlsson, “Opportunistic communication and human mobility”, *Mobile Computing, IEEE Transactions on*, vol. 13, no. 7, pp. 1597–1610, 2014, ISSN: 1536-1233. DOI: `10.1109/TMC.2013.160`.

[19] Ó. R. Helgason and K. V. Jónsson, “Opportunistic networking in OMNeT++”, in *Proc. SIMUTools, OMNeT++ workshop*, Marseille, France, 2008.

Paper G

UrbanCount: Mobile Crowd Counting in Urban Environments

Peter Danielis, Sylvia T. Kouyoumdjieva, and Gunnar Karlsson

In Proceedings of the 8th IEEE Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), pp. 640-648,
`https://doi.org/10.1109/IEMCON.2017.8117189`*, ISBN: 978-1-5386-3371-7,*
October 2017 (Best Paper Award)

UrbanCount: Mobile Crowd Counting in Urban Environments

Peter Danielis*, Sylvia T. Kouyoumdjieva† and Gunnar Karlsson†
* Faculty of Computer Science and Electrical Engineering, University of Rostock, Germany
† ACCESS Linnaeus Center, School of Electrical Engineering, KTH Royal Institute of Technology, Sweden
E-mail: peter.danielis@uni-rostock.de,{stkou,gk}@kth.se

Abstract

Surveillance, management and estimation of spontaneous crowd formations in urban environments, e.g., during open-air festivals or rush hours, are necessary measures for city administration. Most solutions that implement these measures however require additional costly hardware installations (e.g., installation of observation cameras) and infrastructure support, and often pose privacy concerns. In this work, we present *UrbanCount*, a fully distributed crowd counting protocol for cities with high crowd densities. UrbanCount relies on mobile device-to-device communication to perform crowd estimation. Each node collects crowd size estimates from other participants in the system whenever in communication range and immediately integrates these estimates into a local estimate. The objective of UrbanCount is to produce a precise mapping of the local estimate to the anticipated global result while preserving node privacy. We evaluate the proposed protocol via extensive trace-driven simulations of synthetic *and* realistic mobility models. Furthermore, we investigate the dependency between accuracy and density, and demonstrate that in dense environments the local estimate does not deviate by more than 2% for synthetic and 7% for realistic scenarios.

Index terms — Crowd counting, device-to-device communication, mobile networks.

1 Introduction

Crowds are integral parts of cities. As a result, surveillance, management and estimation of crowd densities and sizes are of high importance for city administration, as well as during disaster management. In this work, we focus on the particular case of *crowd counting*, i.e., estimating the size of dense crowds with hundreds to thousands of people. Crowd counting finds application in the context of urban planning for smart cities, especially in scenarios where there is no overall knowledge about the number of people that are located in an area, such as inside a subway station. The most common approach for crowd counting today is the utilization of observation cameras. However, this approach has a number of drawbacks: (1) it requires line-of-sight for operation; (2) it is subject to quality degradation in the case of overlapping objects or changing environmental conditions; and (3) it poses privacy concerns. Other approaches rely on central entities, use measurements obtained from access points or dedicated infrastructure, or utilize different information acquired by human-carried devices for crowd counting (see Section 6).

In this work, we present *UrbanCount*, a fully distributed protocol for mobile crowd counting in urban areas. UrbanCount relies on mobile device-to-device communication to perform accurate crowd estimation and it does not require infrastructure support. UrbanCount allows nodes (i.e., pedestrians equipped with mobile devices) to calculate a local estimate of the *current* number of people in a crowd. Observe that a crowd is often only *intermittently connected*, as only a subset of nodes in the area are in direct communication range of one another at a given time. Furthermore, a crowd is characterized by *churn*, i.e., nodes arrive to an area, stay in it for a while, and then leave. Thus, the local estimate of the crowd size needs to be constantly updated to reflect these changes. Updates occur when nodes in direct communication range exchange crowd size information. Further, local estimates are immediately updated whenever nodes are in range by using any wireless communication interface to receive information.

However, a particular crowd density may be necessary to achieve an accurate crowd size estimate. Consequently, the two most important questions we answer in this paper are:

- What is the minimum required crowd density which enables distributed crowd counting?
- What is the accuracy of the estimate whenever distributed crowd counting is possible?

The main contributions are:

- We present UrbanCount, a fully distributed privacy-preserving protocol for crowd counting via device-to-device communication, which is suitable for operation in urban environments with high node mobility and churn.
- We evaluate UrbanCount via extensive trace-driven simulations of synthetic and realistic mobility models. For densities above 0.1 nodes/m^2, distributed crowd counting produces precise estimates under realistic mobility.
- When the density is sufficient, we show that UrbanCount exhibits the following characteristics: (1) it is shown to be scalable for crowds with up to 2400 people and (2) it achieves high accuracy of at least 98% for synthetic and 93% for realistic mobility scenarios.

The remainder of this paper is organized as follows: We introduce UrbanCount in Section 2. In Section 3, we outline the evaluation scenarios, and in Section 4 and 5 we present results from synthetic and realistic pedestrian mobility scenarios. In Section 6, we compare our approach with the state-of-the-art. Finally, we conclude the study in Section 7, and present directions for future work.

2 UrbanCount: A Distributed Protocol for Crowd Counting

UrbanCount works fully decentralized in such a way that each node is expected to calculate an accurate estimate of the crowd size. To achieve this objective, mobile nodes periodically announce data comprising all nodes that they have obtained knowledge of (shared nodes) and those who have left (left nodes) and thereby the information spreads epidemically. Own data is continuously updated with received data from other nodes in vicinity in order to compute the current crowd size from the difference of the number of shared and left nodes. As crowds are characterized by churn, i.e., nodes entering and leaving the area, each node needs to inform others in its transmission range whenever it is about to exit an area. Note that if a node begins to announce its intention to leave too early or too late, this may lead to wrongful crowd estimations. We here refer to the time interval a node begins to announce its intention to leave as a *farewell time*. We assume a node to have knowledge about a map of the area where it is located (map acquisition is out of scope and left for future studies). The farewell time then defines the boundary region of this area, in which a node starts to announce its intention to leave.

To avoid considering the same participants multiple times, shared and left nodes could be stored in bit vectors whereby each bit position refers to a specific node identity. All bit positions except the one corresponding to the own ID are initialized with 0s. Other bit positions are set to 1 once knowledge about shared or left nodes have been obtained from another contact. However, as pointed out in [1] storing crowd size estimates as a binary bit vector does not scale well as the size of the bit vector grows linearly with the crowd size.

2.1 Introduction to D-GAP compression

To solve the problem of a linearly growing bit vector, in this work we apply *D-GAP compression* [2], which has been shown to be beneficial in [3]. D-GAP compression provides a compressed representation of bit vectors in the form of integer vectors (later referred to as D-GAP vectors) and can be treated as a specialized variant of run length encoding. Each integer is a representation of the number of consecutive 0s or 1s that are part of the bit vector. The first bit of the D-GAP thereby determines if an integer represents a sequence of 0s or 1s. A leading bit of 0 shows that the first integer corresponds to a number of consecutive 0s, followed by a number of consecutive 1s and so on; a leading bit of 1 indicates the opposite behavior. An example of converting a 14-bit vector to a D-GAP vector is depicted in Figure 1. We calculate the total number of bits required

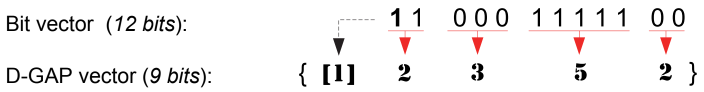


Figure 1: An exemplary D-GAP compression. A 14-bit vector is converted into a D-GAP vector, represented by 12 bits. The leading bit in the D-GAP vector indicates if the vector starts with 0s or 1s.

for representing a D-GAP vector as $\mathrm{N(DGAP)} = \sum_{i=1}^{n}(\lfloor \log_2 d_i \rfloor + 1)$ where d_i is the integer representation at position i of the D-GAP vector, and n is the size of the vector. Hence, the binary representation of this D-GAP vector consists of 12 bits leading to a compression ratio of 2/14=14%. The main advantage of D-GAP in front of bit vectors is however seen at large numbers. Imagine a crowd consisting of 10,000 nodes, whose identities need to be stored. In the best case scenario, storing 10,000 consecutive 1s in a D-GAP vector results in a total of 14 bits as compared to 10,000 bits for a regular bit vector, thus leading to a compression ration of 99.9%. In the worst-case scenario, a D-GAP vector of alternating 1s and 0s would lead to no compression, however, this scenario is considered as very unlikely. Small vector sizes are of paramount importance for fitting information into packets exchanged over an unreliable wireless link during a short contact (see Section 2.3).

2.2 Operations on D-GAP vectors

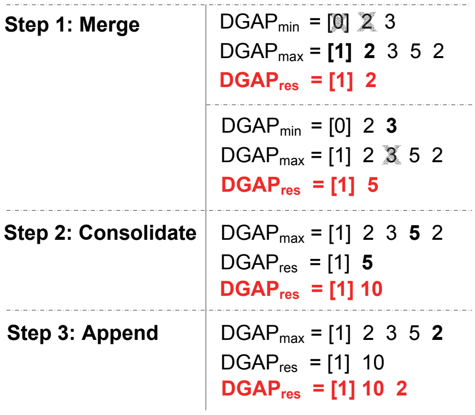


Figure 2: Merging, consolidating, and appending D-GAP vectors. Bold integers refer to positions under consideration, crossed integers remain unconsidered, and the resulting vector is depicted in red.

We define three operations that can be executed on two D-GAP vectors of any lengths: `merge`, `consolidate`, and `append`. Let us assume vectors of different lengths, DGAP_{min} and DGAP_{max}, denoting the shorter and the longer vector, respectively. Observe that the vector length L corresponds to the sum of all D-GAP integers. For instance, DGAP_{min} could be a node's local D-GAP vector and DGAP_{max} could be a received D-GAP vector, or vice versa.

- The `merge` operation merges DGAP_{min} and $\mathrm{DGAP}_{max}[1{:}L(\mathrm{DGAP}_{min})]$ into a resulting vector DGAP_{res}. The `merge` operation is executed iteratively until the end of DGAP_{min} and corresponds to a bitwise logical OR on the equivalent uncompressed bit vectors.

```
{
  "message" : {
    "node-id" : "0x01:0b:84:7e:57:05",
    "urbancount" : [ {
      "crowd" : {
        "crowd-id" : "5",
        "subject" : "crowd counting subway",
        "shared" : [0 23 1],
        "left" : [],
        "updated" : "2016-07-29T10:47:02Z"
      }
    } ]
  }
}
```

Listing 2: A sample UrbanCount message in the JSON format.

- The `consolidate` operation combines the last position of DGAP$_{res}$ with the next position of DGAP$_{max}$ after the `merge` operation has been executed. The `consolidate` operation is solely executed if the integers at these two positions have the same bit value.
- The `append` operation appends the remainder of DGAP$_{max}$ to DGAP$_{res}$.

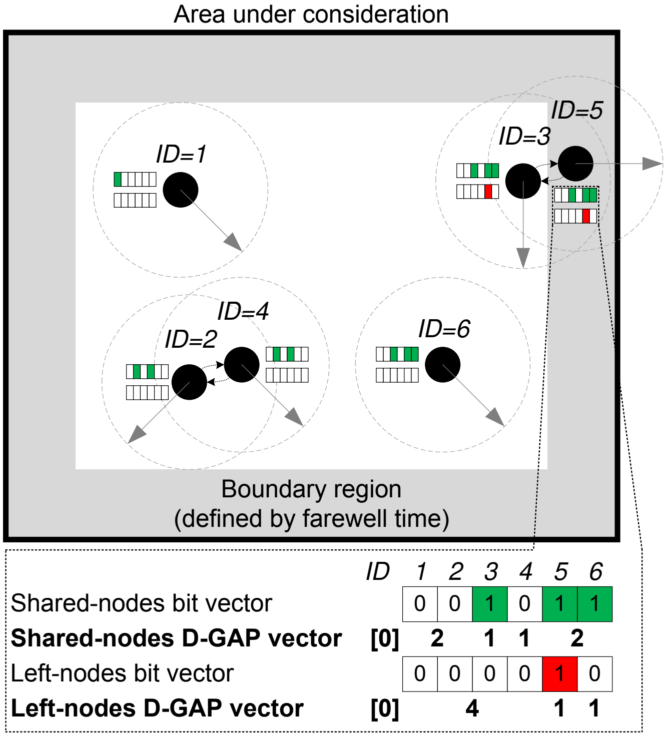
Note that for two input vectors of equal lengths, only the `merge` operation is executed. Figure 2 depicts an example.

2.3 Functional principles of UrbanCount

In UrbanCount, information is represented by two D-GAP vectors: a *shared-nodes vector* and a *left-nodes vector*. To demonstrate the functionality of UrbanCount, we will use a simple example of six mobile nodes, see Figure 3. Upon entry into the area, a node inserts itself into its shared-nodes vector and its left-nodes vector remains empty. For privacy preservation reasons, the position, in which information is stored in each of the D-GAP vectors, is determined by a cryptographic hash function such as MD5, which is calculated over a unique identifier, e.g., the node's MAC address. For instance, for node 1 in Figure 3 the cryptographic hash function returns a position of 1, thus its shared-nodes vector is initialized with [1 1 5].

While in the area, each node broadcasts a beacon at regular intervals, which comprises its shared-nodes vector and its left-nodes vector to other nodes in its transmission range. A sample UrbanCount beacon message is shown in Listing 2. Upon reception of a beacon from another contact, a node updates its local shared-nodes vector and its left-nodes vector. In Figure 3, e.g., nodes 2 and 4, as well as nodes 3 and 5, have already exchanged their vectors and now have identical views on the population in the area.

The executed operations to combine the local with the received D-GAP vectors are outlined in Algorithm 1. Note that the algorithm is executed on both the shared-nodes vector and the left-nodes vector. First a node extracts the received information and



	ID	1	2	3	4	5	6
Shared-nodes bit vector		0	0	1	0	1	1
Shared-nodes D-GAP vector	**[0]**	**2**		**1**	**1**	**2**	
Left-nodes bit vector		0	0	0	0	1	0
Left-nodes D-GAP vector	**[0]**	**4**				**1**	**1**

Figure 3: An example of six mobile nodes executing the UrbanCount protocol. Information is represented by two D-GAP vectors and the respective bit vectors are solely depicted for clarity.

checks in a local database whether the received vector from peer i, $\mathrm{DGAP}_{rec}^{(i)}$, has changed (line 4). This check is currently done by looking up the `node-id` in the local database (a simple table containing the `node-id` and `updated` fields). If the `node-id` is found, there has been prior communication with this peer and the advertised `updated` field value is compared to the previously registered `updated` field value. If they do not differ, the local vector DGAP_{loc} stays unchanged. Otherwise, UrbanCount initializes DGAP_{min} and DGAP_{max} (lines 5-6), defined in Section 2.2, and consecutively executes the operations `merge` (lines 7-11), `consolidate` (lines 12-16) and `append` (line 17) in order to update the local estimate (line 18). Thus, the shared-nodes vector and the left-nodes vector contain cumulative information of all nodes that have been shared or have been seen leaving over time. From the difference of both numbers, a node can compute the current local crowd size estimate. Note that correlating nodes hash values and their identities is impossible by third party entities such as central collection points or compromised participants.

Eventually, a node inserts itself to its left-nodes vector when it reaches the beginning of its farewell time interval. (More details on how the farewell time is calculated are presented later in the paper.) For instance, in Figure 3 node 5 has already entered the boundary region, and now announces a left-nodes vector of [0 4 1 1].

Algorithm 1 Operations on D-GAP vectors

```
$\mathrm{DGAP}^{(i)}_{rcv} \leftarrow$ received DGAP vector from node $i$
$\mathrm{DGAP}_{loc} \leftarrow$ local DGAP vector
$\mathrm{DGAP}_{res} \leftarrow$ resulting DGAP vector
if $\mathrm{DGAP}^{(i)}_{rcv}$ changed since last beacon from node $i$ then
    $\mathrm{DGAP}_{min} = \min(\mathrm{DGAP}_{loc}, \mathrm{DGAP}^{(i)}_{rcv})$
    $\mathrm{DGAP}_{max} = \max(\mathrm{DGAP}_{loc}, \mathrm{DGAP}^{(i)}_{rcv})$
    while ! $\mathrm{DGAP}_{min}.end()$ do
        $\mathrm{DGAP}_{res} = \textsc{Merge}(\mathrm{DGAP}_{min}, \mathrm{DGAP}_{max})$
        $rpos \leftarrow$ end position in $\mathrm{DGAP}_{res}$
        $mpos \leftarrow$ current position in $\mathrm{DGAP}_{max}$
    end while
    $\mathrm{state}_{res} = (rpos \mod 2)\ xor\ \mathrm{DGAP}_{res}[0]$
    $\mathrm{state}_{max} = (mpos \mod 2)\ xor\ \mathrm{DGAP}_{max}[0]$
    if $\mathrm{state}_{res}\ ==\ \mathrm{state}_{max}$ then
        $\mathrm{DGAP}_{res} = \textsc{Consolidate}(\mathrm{DGAP}_{res}[rpos],$
[10] $\mathrm{DGAP}_{max}[mpos])$
    end if
    $\mathrm{DGAP}_{res} = \textsc{Append}(\mathrm{DGAP}_{res}, \mathrm{DGAP}_{max}[mpos : \mathrm{end}])$
    $\mathrm{DGAP}_{loc} \leftarrow \mathrm{DGAP}_{res}$
end if
```

The information dissemination, i.e., shared-nodes and left-nodes vector, between *any* two nodes depends on their mobility *and* their previously obtained knowledge about other participants. The system could be described as an epidemic model, in which multiple simultaneous infections are possible [4]; the analogy would be that infections correspond to the particular nodes a node becomes aware of over time. As shown in [4], such a model has a very high complexity, and is intractable in the general case. Therefore, we renounce modelling the system and rather evaluate it by means of simulations.

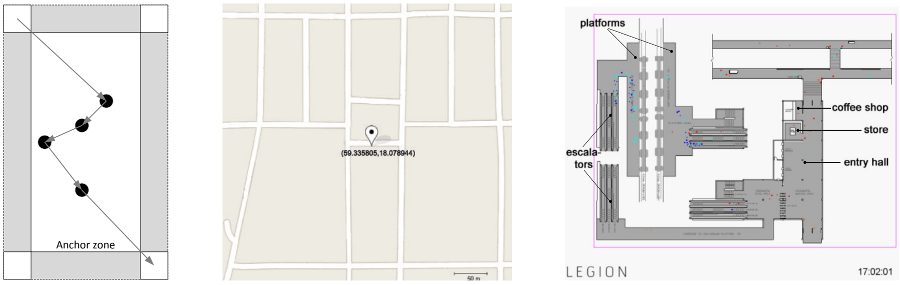
3 Evaluation Scenarios

In this section, we introduce the mobility scenarios as well as the simulation setup and the investigated performance metrics. Mobility scenarios are among other things characterized by a node arrival rate λ, with which nodes arrive in an area A. Nodes stay in the area with some lifetime T.

3.1 Mobility scenarios

3.1.1 City square model [5]

This open random waypoint model describes the movement of nodes in a confined area such as a city square. The main difference between this form of random waypoint and the traditional one is the varying population. A node arrives to any of the four entry points according to a Poisson process, and immediately moves to a random point in the anchor zone. At a waypoint, a node may choose to exit the area with probability p, or



(a) (b) (c)

Figure 4: Urban scenarios: (a) city square mobility model with random waypoint node movement in the anchor zone, (b) a grid of streets representing the Östermalm scenario, and (c) a two-level subway station.

to move to another randomly chosen waypoint inside the area. In our setup, nodes move in a rectangular area of size 5000 m^2, Figure 4 (a). Nodes move at a speed of 1.3 m/s without pausing between two consecutive waypoints.

3.1.2 Realistic pedestrian mobility

In order to realistically recreate pedestrian mobility, we use the Walkers traces [6] captured in Legion Studio [7], a commercial simulator initially developed for designing and dimensioning large-scale spaces via simulation of pedestrian behaviors. Its multi-agent pedestrian model is based on advanced analytical and empirical models which have been calibrated by measurement studies. Each simulation run results in a trace file, containing a snapshot of the positions of all nodes in the system every 0.6 s. Fig. 4 (b) and 4 (c) present the realistic scenarios considered in our evaluation: an outdoor urban scenario, modeling a city area (which we later refer to as the Östermalm scenario), and an indoor scenario, recreating a two-level subway station. The Östermalm scenario consists of a grid of interconnected streets. Fourteen passages connect the observed area to the outside world. The active area, i.e., the total surface of the streets, is 5872 m^2. The nodes are constantly moving, hence the scenario can be characterized as a high mobility scenario. The Subway station has train platforms connected via escalators to the entry-level. Nodes arrive on foot from any of five entries, or when a train arrives at the platform. The train arrivals create burstiness in the node arrivals and departures. Nodes congregate while waiting for a train at one of the platforms, or while taking a break in the store or the coffee shop at the entry level. The active area is 1921 m^2.

3.2 Simulation setup

In our evaluation scenarios, we assume that (1) all nodes carry devices and (2) all are participating in the crowd counting. Scenarios, in which not all nodes participate, are out of scope and left for further investigation. For the evaluation, we use an implementation of an opportunistic content distribution system in the OMNeT++ modeling framework MiXiM [8]. Each simulation run is executed in synchronous rounds of 0.6 s which corresponds to the granularity of the mobility traces we use. Nodes broadcast their shared-nodes and left-nodes vector at the beginning of each round[1]. To avoid collisions on the wireless medium, the broadcast transmission of each node in each round is distributed uniformly at random $\mathcal{U}(0, 0.5)$ s. The transmission range is set to 10 m.

3.3 Performance metrics

We focus on evaluating the following performance metrics: *Accuracy* (Δ): The deviation is a measure of the accuracy of the UrbanCount protocol, i.e., it shows how close the local estimate of a node is to the anticipated global result. Let $\bar{x}$ be the local estimate (cf. Section 2.3) and x the actual global result. Then, the deviation is calculated as:

$$\Delta = \left| \frac{\bar{x} - x}{x} \right| \tag{1}$$

Processed and useful broadcasts: The share of processed and useful broadcasts indicates the dynamism of information dissemination. Processed broadcasts are broadcasts received from a node that has not been encountered previously or a known node that announces an updated timestamp in its `updated` field. Useful broadcasts are received broadcasts that contribute with new data to the local estimate of a node. We calculate the share of processed and useful broadcasts at the application layer with respect to the total number of broadcasts received by a node. Observe that we do not aim at optimizing broadcast intervals in this work.

4 Simulation Results: City Square Model

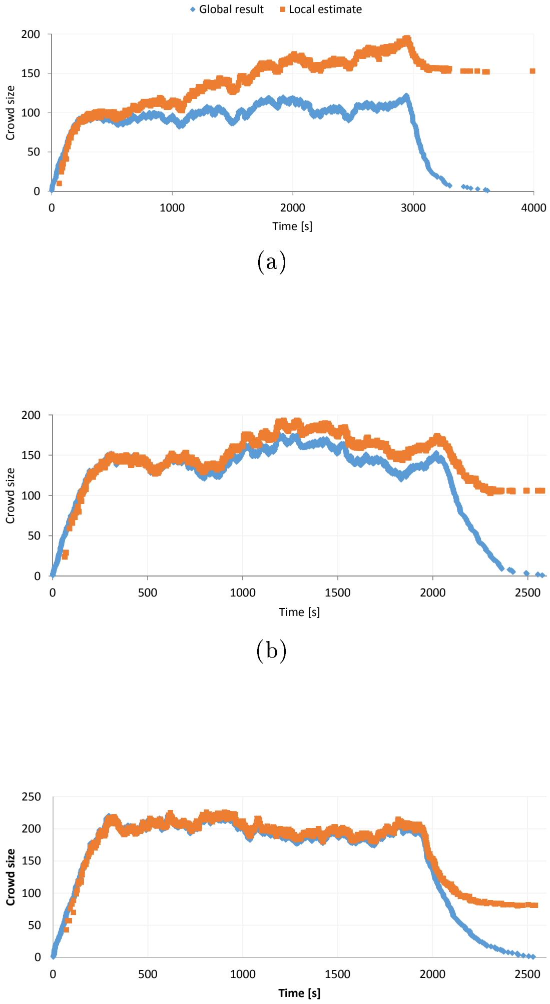
In this section, we evaluate the performance of UrbanCount on the city square model, Figure 4 (a), with respect to different arrival rates as well as varying farewell times.

4.1 Effect of arrival rate

We first evaluate the effect of the arrival rate λ on the deviation Δ; λ here is the total arrival rate into the observed area, i.e., the arrival rate for *all four entry points* of the city square. We set $\lambda = \{0.05..4\}$ nodes/s. Scenarios for which $\lambda < 0.05$ nodes/s are not considered, as they are too sparse and result in little to no node encounters. Scenarios with $\lambda > 4$ are not considered either since they do not improve the observed accuracy, i.e, convergence is already guaranteed. High λ values were chosen to represent dense scenarios in order to show that the UrbanCount protocol even scales and works in these cases. The farewell time is set to α·ICT where α is a scaling factor and ICT denotes the average inter-contact time for *all* nodes in the system (we release this assumption in Section 5.3 and allow nodes to individually measure the ICT). ICT is measured as the time that elapses from the beginning of one contact to the beginning of the next one [9].

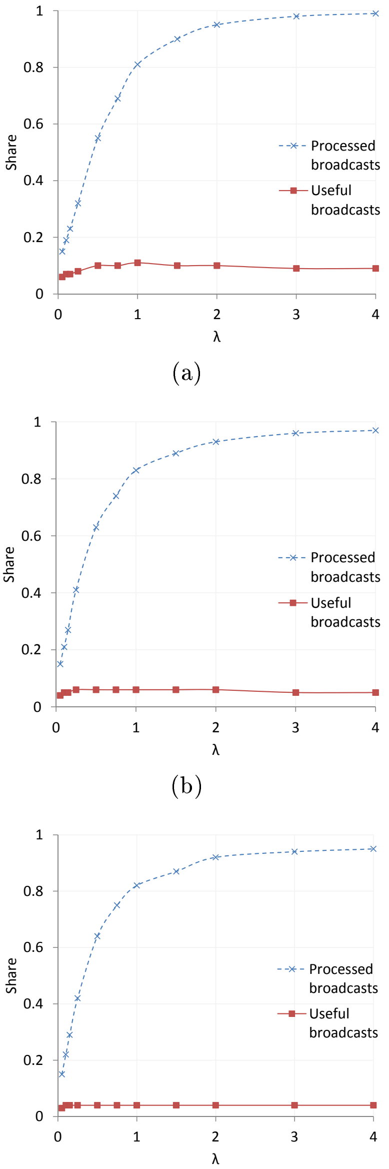
---

[1] As proof-of-concept, we implemented apps for message broadcast between smartphones with Android and iOS using Bluetooth LE and Wi-Fi (Wi-Fi P2P on Android).



(a) (b) (c)

Figure 5: Local crowd size estimates of all nodes vs. global result for the city square scenario with an average node lifetime $T = 200$ s: (a) $\lambda = 0.5$ nodes/s, (b) $\lambda = 0.75$ nodes/s, and (c) $\lambda = 1.0$ nodes/s.



(a) (b) (c)

Figure 6: Share of processed and useful broadcasts for the city square scenario with an average node lifetime (a) T = 200 s, (b) T = 400 s, and (c) T = 600 s.

We here set $\alpha = 2$ and explore different values of α in the following section. We evaluate the performance for three different values of the average lifetime of nodes in the system, $T = \{200, 400, 600\}$ s. The average node number is up to 2400 nodes. Table 1 shows the estimated mean density for each configuration for a subset of the evaluated arrival rates λ, in accordance to Little's law. If not stated otherwise, Δ is calculated across 1000 nodes once the system reaches steady state.

Table 1: Crowd densities for different lifetimes T and arrival rates λ following Little's law with A = 5000 m^s.

	λ	0.5	0.75	1	1.5	2	3	4
T=200s	ρ	0.02	0.03	0.04	0.06	0.08	0.12	0.16
T=400s	ρ	0.04	0.06	0.08	0.12	0.16	0.24	0.32
T=600s	ρ	0.06	0.09	0.12	0.18	0.24	0.36	0.48

Figure 5 shows the local node estimates of the crowd size vs. the global result over time for $\lambda = \{0.5, 0.75, 1\}$ nodes/s and $T = 200$ s. Note that the first estimates are recorded around 50 s after the beginning of the simulation; the reason is that we record the crowd size estimate only when nodes leave the area. In sparser scenarios, Figure 5 (a) and 5 (b), the local estimate produces an overestimate of the crowd size, as there is no sufficient information about the nodes that leave the system. Increasing λ leads to a more accurate local node estimate as information spreads quickly due to the larger amount of contacts that nodes experience in the area. However, regardless the node density in the area, the local estimate is unable to follow the global result at the end of the simulation. The reason for this behavior is that as nodes congregate at one of the four exit points at the end of the simulation run, they are unable to communicate with nodes at the other exit points due to the limit in their communication range. Note that this behavior is particular to the scenario at hand, as we will see in Section 5.

Table 2: Average ICT and Δ for the city square scenario for different node life times and arrival rates λ.

	λ nodes/s	Avg. ICT [s]	Avg. Δ (final estimate)	Avg. Δ (over time)
T=200s	0.5	1.9	0.44±0.15	0.46±0.15
	1	1.2	0.02±0.01	0.02±0.07
T=400s	0.5	1.2	0.17±0.07	0.17±0.09
	1	0.8	0.01±0.01	0.01±0.05
T=600s	0.5	1.0	0.07±0.02	0.08±0.05
	1	≤0.75	0.01±0.01	0.01±0.04

Table 2 provides a comparison between the values of Δ recorded when a node leaves the area and over time, i.e., while a node is inside the area. We only show the comparison for two values of $\lambda = \{0.5, 1.0\}$ nodes/s; for $\lambda < 0.5$ nodes/s the deviation is too high and produces erroneous results, and for $\lambda \geq 1.0$ nodes/s the deviation is always below 2%. As it can be seen from Table 2, the deviation calculated over time is only slightly higher than the deviation calculated at the end of the simulation. Thus, we can conclude that the local estimates can be regarded as stable at any point in time for dense scenarios.

Figure 6 shows how much of the information a node receives from its neighbors is processed or useful for updating its local estimate. We note that the share of processed broadcasts increases rapidly as the density increases across all node lifetimes. For low crowd densities, most processed broadcasts result in useful broadcasts. As density increases, the share of useful broadcasts does not exceed 10%. This indicates that information spreading in the system is efficient, which is also confirmed by the measured deviation ($\Delta < 0.01$).

Figure 7 shows that deviation decreases exponentially with the increase of the arrival rate regardless of the average node lifetime in the system. (Curve fitting shows that the trend can be described as a sum of two exponential distributions, $\Delta(\lambda) = a \cdot e^{b\lambda} + c \cdot e^{d\lambda}$; parameters as well as the respective coefficients of determination R^2 are shown in the figure.) We see that for $\lambda \geq 1.0$ nodes/s or densities higher than or equal to 0.04 nodes/m^2 (Table 1), deviation is reduced below 2%.

4.2 Effect of farewell time

We further investigated the impact of different farewell times on Δ for two different values $\lambda = \{0.5, 1\}$ nodes/s. We remind the reader that the farewell time is calculated as α·ICT. The average ICT values are given in Table 2. For each configuration we vary the scaling factor $\alpha = 2^n$ where $n = 0.5..10$. The results are shown in Figure 8 for different average node lifetime of $T = \{200, 400, 600\}$ s. We see that for both values of λ, the deviation first decreases with increase of α until it reaches a minimum. We note that this trend is more visible for $\lambda = 0.5$ nodes/s; in this case the deviation is below 6% for $n = [1..3]$ while for $\lambda = 1.0$ nodes/s, $n = [2..3]$. We also note that the deviation rapidly increases whenever the scaling factor exceeds a certain value. The reason is that whenever α is large, nodes begin to announce that they are leaving the system much earlier than their actual leaving time. This in turn leads to wrongful estimation of the crowd count. Thus, setting a correct value for the scaling factor α is of high importance for the proper functioning of our crowd counting protocol. We recommend the value to be kept relatively low, and not to exceed 8·ICT. Note that we do not set the farewell time to the minimum values, as this may lead to the underestimation of leaving nodes.

5 Simulation Results: Urban Scenarios

In this section, we first evaluate the performance of UrbanCount for realistic urban scenarios with respect to different arrival rates and various farewell times. We further

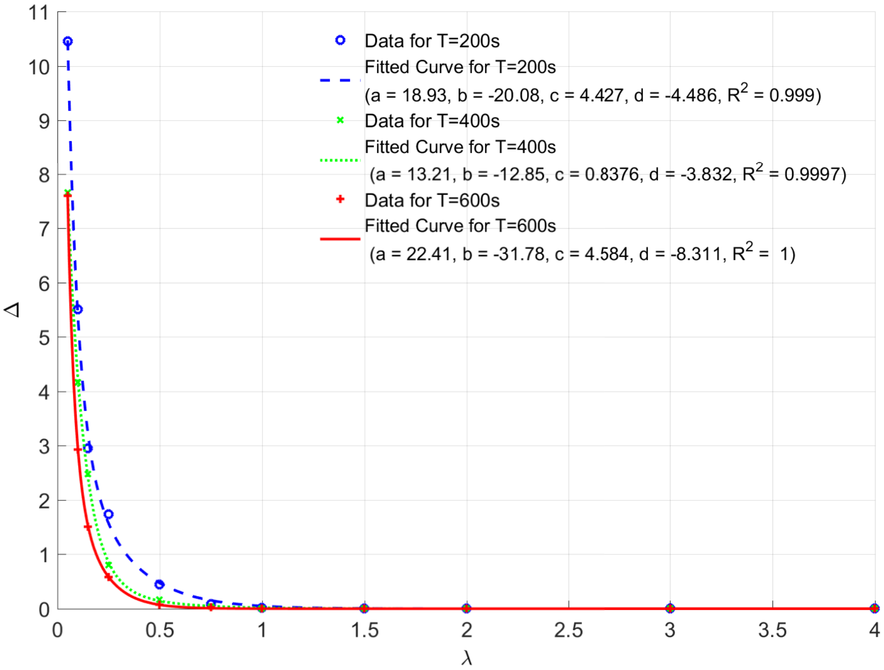


Figure 7: Average deviation Δ over different arrival rates λ for the city square scenario. $\Delta = 1$ denotes an estimate deviating by 100% from the global result.

show a fully distributed solution, in which nodes individually evaluate their perceived inter-contact times.

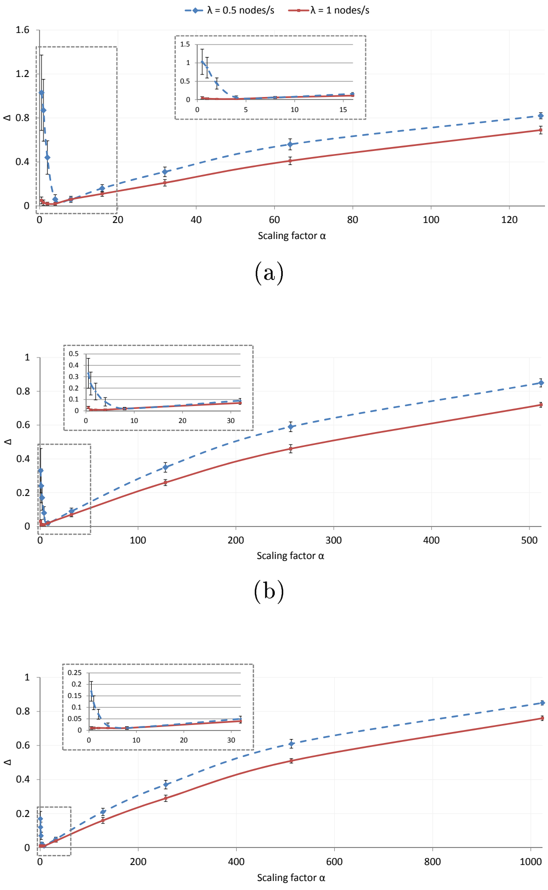
5.1 Effect of arrival rate

We first present results for the Östermalm scenario. Nodes arrive to the scenario with an arrival rate $\lambda_1 = \{0.01..0.3\}$ nodes/s from each of the fourteen entry points to the area, see Figure 4 (b). The average node number is up to 1035 nodes. Below, we present results for $\lambda = 14 \cdot \lambda_1$. Again, we set the farewell time to $\alpha\cdot$ ICT with $\alpha = 2$; a complete study on the effect of α is provided in Section 5.2.

Figure 9 shows the local estimates vs. global result over time for $\lambda = \{0.42,\ 0.98,\ 2.1\}$ nodes/s. The results confirm that in sparser scenarios, Figure 9 (a) and 9 (b), the crowd estimate is often overestimated due to insufficient information spreading, however as λ increases, the local estimate approaches the global result.

Table 3 summarizes the performance of UrbanCount in the Östermalm scenario across different arrival rates. Similar as before, the share of useful broadcasts is kept below 10% across all values of λ, however the total amount of processed broadcasts is much higher.

Figure 10 shows the dependency of deviation with respect to the arrival rate. Although the scenario has a higher complexity than the city square model, the deviation still decreases exponentially with increase of λ (parameters of the fitted curve are provided in

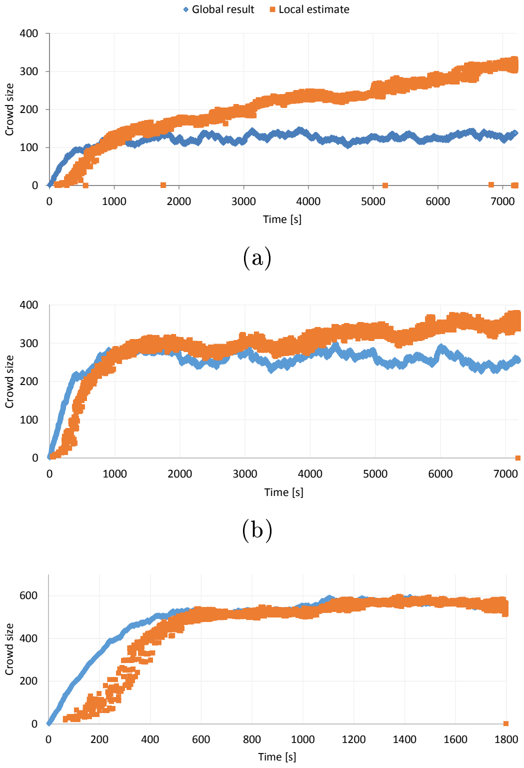


(a)

(b)

(c)

Figure 8: Average deviation of local crowd size estimates of all nodes from global result with different farewell times for the city square scenario with an average node lifetime: (a) T = 200 s, (b) T = 400 s, and (c) T = 600 s. Δ = 1 denotes an estimate which deviates by 100% from the global result.



(a)

(b)

(c)

Figure 9: Local crowd size estimates of all nodes vs. global result for the Östermalm scenario: (a) $\lambda = 0.42$ nodes/s, (b) $\lambda = 0.98$ nodes/s, and (c) $\lambda = 2.1$ nodes/s.

the figure). We observe that densities greater or equal to 0.1 nodes/m^2 enable distributed crowd counting with a deviation of at most 5%.

Table 3: Average ICT and individual ICT (i-ICT) and share of processed and useful broadcasts (BCs) for the Östermalm scenario with different arrival rates λ.

λ [nodes/s]	0.14	0.28	0.42	0.7	0.98	2.1	4.2
Avg. ICT [s]	38.7	20.3	13.6	8.9	6.8	3.5	2.1
Avg. i-ICT [s]	39.8	21.0	13.9	9.0	6.9	3.6	2.1
Proc. BCs [%]	11%	13%	15%	18%	20%	33%	61%
Useful BCs [%]	5%	5%	6%	6%	6%	7%	9%

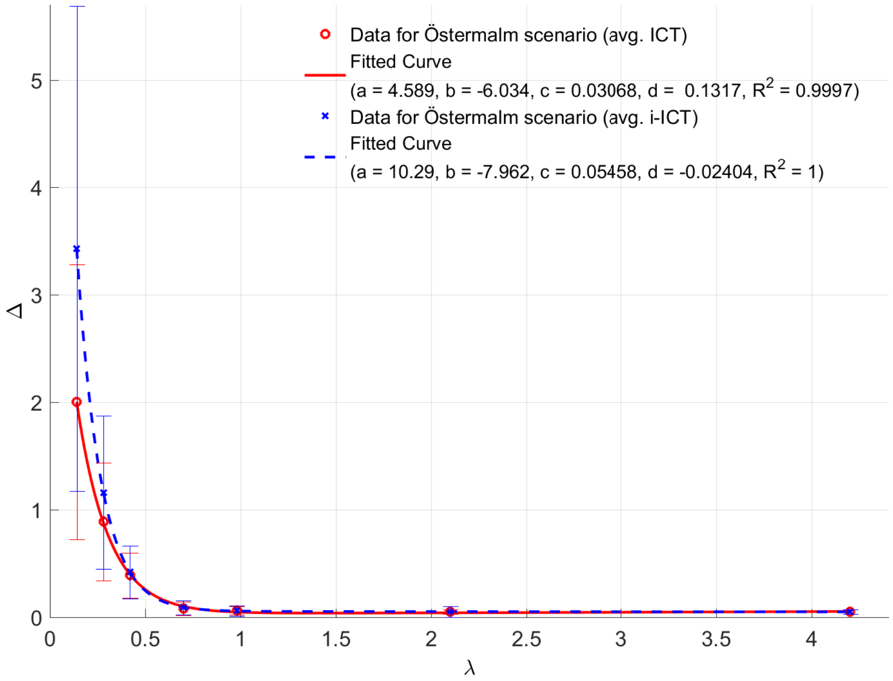


Figure 10: Average deviation Δ for different arrival rates λ for the Östermalm scenario. $\Delta = 1$ denotes an estimate deviating by 100% from the global result.

5.2 Effect of farewell time

We further study the effect of farewell times for the Östermalm scenario for $\lambda = \{0.7, 2.1\}$ nodes/s. Again, farewell times are calculated as α·ICT, where the average ICT values are given in Table 3. As the mean sojourn time of nodes is different across configurations ($T = 342$ s for $\lambda = 0.7$ nodes/s and $T = 293$ s for $\lambda = 2.1$ nodes/s), the maximum scaling factor $\alpha = 2^n$ is calculated for $n = \{6, 7\}$, respectively. Scaling factors higher than these values would result in nodes announcing their leaving times from the moment they enter in the area, and are thus not considered. The results in Figure 11 confirm that even in complex scenarios, low values of α lead to a deviation below 10%. Thus, for realistic

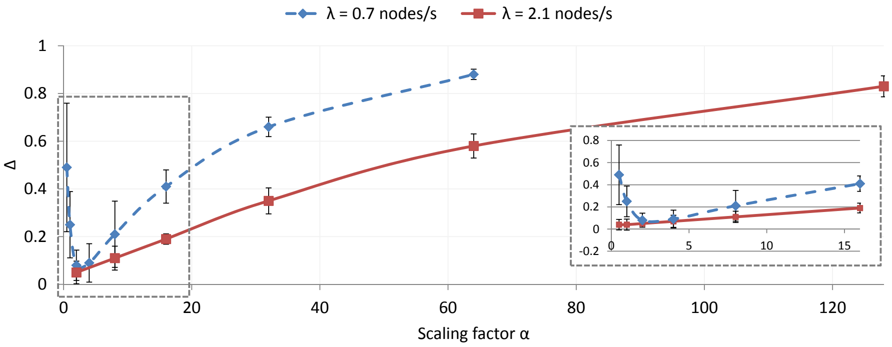
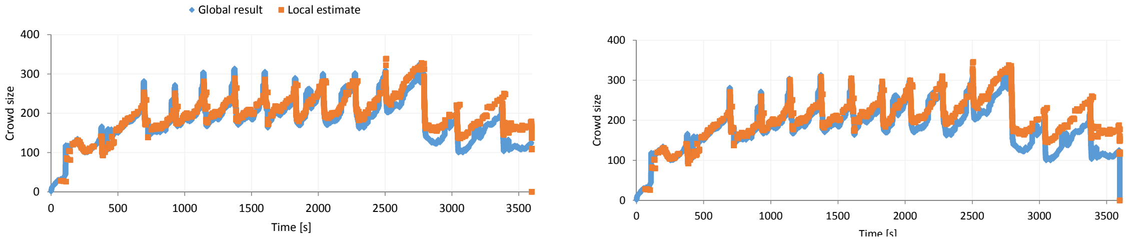


Figure 11: Average deviation of local crowd size estimates of all nodes from global result with different farewell times for the Östermalm scenario. $\Delta = 1$ denotes an estimate which deviates by 100% from the global result.

scenarios the farewell time should be set to no more than 4·ICT.

5.3 Effect of individual inter-contact times

Until now, nodes used a globally known predefined average ICT to adjust their farewell times. In order to achieve a fully distributed solution for UrbanCount, in this section we allow nodes to *individually* measure the inter-contact times they experience throughout their lifetime in the area. We evaluate the effect of individual ICT for the Östermalm and Subway scenarios. We note that as nodes take different routes, they experience different lifetimes and thus varying contact rates. However, Table 3 shows that the average ICT and the average individual ICT do not differ significantly for the Östermalm scenario. Moreover, as λ increases, the difference between measured and average ICT becomes negligible.



(a)

(b)

Figure 12: Local crowd size estimates of all nodes vs. global result for the Subway scenario, (a) with nodes utilizing an average ICT value and (b) with nodes utilizing individually measured ICT values.

Figure 10 further compares the deviation from the local estimate when nodes utilize an average ICT value and when they individually measure ICT. The results show that the deviation does not change significantly, however individual measures always produce slightly worse performance, especially in sparser scenarios. This is due to the fact that nodes with shorter lifetimes do not experience sufficiently high number of contacts to correctly estimate the average ICT value. This leads to even higher deviation from the global result and is further reflected in the higher standard deviations in Figure 10 for small λ.

Finally in Figure 12, we show the performance of UrbanCount in the Subway scenario with an average node number of 205 nodes. The density of the Subway scenario is approximately 0.1 nodes/m^2, which makes it appropriate for our distributed crowd counting protocol. We note that the scenario is more challenging than the Östermalm scenario due to the bursty nature of node arrivals and departures. The results show that UrbanCount is able to correctly estimate the crowd size in the area regardless of whether nodes use average or individually measured ICT values, Figures 12 (a) and 12 (b) respectively. We note that for both configurations, the deviations from the global estimate are no more than 7%.

6 Related Work

Robust crowd counting can be carried out by three different strategies, namely video-based, device-free non-image based, and device-based non-image based recognition. In what follows, we solely compare UrbanCount to device-based non-image based recognition methods due to space constraints.

Device-based non-image based recognition comprises approaches where users carry radio devices, such as RFID tags, mobile phones, and/or sensor nodes, to count or locate objects. In [10] and [11], the authors present a technique for estimating crowd density by using a mobile phone to scan the environment for discoverable Bluetooth devices to analyze crowd conditions in urban environments. The authors acknowledge that their solution needs to recruit volunteers who follow a specific mobility pattern to meet sufficiently many people. They leave for future work how many volunteers need to be recruited and what mobility patterns would be needed. In [12], the authors leverage audio tones for crowd counting. Mobile devices periodically broadcast a bit pattern corresponding to the audio frequencies received by other devices in transmission range. Results show up to 90% accuracy in certain experimental configurations with up to statically deployed 25 Android devices. However, 50% estimation errors are experienced in other scenarios and the counting latency increases significantly with an increasing device number, which makes the proposed algorithm unsuitable for highly dynamic scenarios. An approach for performing concurrent estimations in dense wireless networks with up to 100 nodes is presented in [13], which uses the rendezvous time to capture the density of the neighborhood. Mobility is taken into account but the evaluation is limited to a few moving nodes. The work in [14] proposes to exploit opportunistically smartphones' acoustic sensors in presence of human conversation and motion sensors in absence of any conversational data to determine the number of occupants in a crowded environment. However, their approach is solely suitable for small groups of up to 10 people. UrbanCount, on the other hand, scales well with the number of nodes in an area, and is able to estimate crowds of various sizes. In [15], the authors propose an application for estimating the size of a (mobile) crowd, which operates by epidemically spreading small messages between users. They define a stochastic model to study the operation of the application. However, the model fails to capture changes in scenarios with bursty arrivals and departures, such as in the Subway scenario, as it assumes constant arrival rates. As opposed to their work, UrbanCount is able to correctly estimate the crowd size even in highly dynamic scenarios.

Crowd counting can also be regarded as a distributed aggregation problem with a counting objective. There are two main paradigms to address such aggregation problems, namely restarted and bookkeeping gossip-based and tree-based aggregation. On the one hand, tree-based aggregation protocols perform poorly in dynamic environments with high levels of churn [16]. On the other hand, restarted gossip algorithms lack accuracy, while bookkeeping gossip protocols may be slow to converge, which makes them inappropriate for application in scenarios with high dynamics. UrbanCount however is able to produce accurate estimates in highly dynamic environments.

7 Conclusion

Surveillance, management, and estimation of spontaneous crowd formations in urban environment, e.g., during open-air festivals or rush hours, are necessary measures for city administration. In this work, we presented *UrbanCount*, a fully distributed protocol for crowd counting via device-to-device communication, which is suitable for operation in urban environments with high node mobility and churn. Each node collects crowd size estimates from other participants in the system whenever in direct communication range and immediately integrates these estimates into a local estimate. We evaluated the proposed protocol via extensive trace-driven simulations of synthetic and realistic mobility models, and we answered two main questions: *when* is distributed counting possible, and *how* accurate can it be. We showed that for densities above 0.1 nodes/m^2 distributed crowd counting always produces precise estimates regardless of the area in which mobility occurs. When the density is sufficient, we showed that UrbanCount is able to produce a local estimate with at least 98% accuracy for synthetic and 93% for realistic mobility scenarios. Furthermore, we presented that UrbanCount provides a scalable solution for crowd counting for both indoor and outdoor scenarios. Finally, we showed that the performance of UrbanCount is not affected if the characteristics of mobility in an area are not known in advance. As part of our future work, we plan to evaluate other scenarios in a realistic testbed, including an investigation of interference and path-loss effects as well as communication technology-specific impacts.

Acknowledgment

The authors would like to thank the German Research Foundation (DFG) (research fellowship, GZ: DA 1687/2-1) and Stockholm County Council (SLL) (research grant LS2016-1423) for their financial support.

8 References

[1] D Chronopoulos, “Extreme chaos: Flexible and efficient all-to-all data aggregation for wireless sensor networks”, PhD thesis, TU Delft, Delft University of Technology, 2016.

[2] A. Kuznetsov, *D-gap compression*, 2002. [Online]. Available: `http://bmagic.sourceforge.net/dGap.html`.

[3] P. Danielis, S. T. Kouyoumdjieva, and G. Karlsson, “Divote: A distributed voting protocol for mobile device-to-device communication”, in *2016 28th International Teletraffic Congress (ITC 28)*, vol. 01, 2016, pp. 69–77. DOI: `10.1109/ITC-28.2016.118`.

[4] X.-S. Zhang and K.-F. Cao, “The impact of coinfections and their simultaneous transmission on antigenic diversity and epidemic cycling of infectious diseases”, *BioMed Research International*, vol. 24, 2014.

[5] M. S. Desta, E. Hyytiä, J. Ott, and J. Kangasharju, “Characterizing content sharing properties for mobile users in open city squares”, in *Wireless On-demand Network Systems and Services (WONS), 2013 10th Annual Conference on*, 2013, pp. 147–154. DOI: `10.1109/WONS.2013.6578340`.

[6] S. T. Kouyoumdjieva, Ó. R. Helgason, and G. Karlsson, *CRAWDAD data set kth/walkers (v. 2014-05-05)*, Downloaded from http://crawdad.org/kth/walkers/, May 2014.

[7] *Legion Studio*, `http://www.legion.com/`.

[8] Ó. R. Helgason and K. V. Jónsson, “Opportunistic networking in OMNeT++”, in *Proc. SIMUTools, OMNeT++ workshop*, Marseille, France, 2008.

[9] Ó. Helgason, S. T. Kouyoumdjieva, and G. Karlsson, “Opportunistic communication and human mobility”, *Mobile Computing, IEEE Transactions on*, vol. 13, no. 7, pp. 1597–1610, 2014, ISSN: 1536-1233. DOI: `10.1109/TMC.2013.160`.

[10] J. Weppner and P. Lukowicz, “Collaborative crowd density estimation with mobile phones”, *Proc. of ACM PhoneSense*, 2011.

[11] J. Weppner, P. Lukowicz, U. Blanke, and G. Tröster, “Participatory bluetooth scans serving as urban crowd probes”, *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4196–4206, 2014, ISSN: 1530-437X.

[12] P. G. Kannan, S. P. Venkatagiri, M. C. Chan, A. L. Ananda, and L.-S. Peh, “Low cost crowd counting using audio tones”, in *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems*, Toronto, Ontario, Canada: ACM, 2012, pp. 155–168, ISBN: 978-1-4503-1169-4.

[13] M. Cattani, M. Zuniga, A. Loukas, and K. Langendoen, “Lightweight neighborhood cardinality estimation in dynamic wireless networks”, in *Proceedings of the 13th international symposium on Information processing in sensor networks*, IEEE Press, 2014, pp. 179–189.

[14] M. A. A. H. Khan, H. M. S. Hossain, and N. Roy, “Sensepresence: Infrastructure-less occupancy detection for opportunistic sensing applications”, in *Mobile Data Management (MDM), 2015 16th IEEE International Conference on*, vol. 2, 2015, pp. 56–61.

[15] L. Pajevic and G. Karlsson, “Characterizing opportunistic communication with churn for crowd-counting”, in *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2015 IEEE 16th International Symposium on a*, 2015, pp. 1–6.

[16] L. Nyers and M. Jelasity, “A comparative study of spanning tree and gossip protocols for aggregation”, *Concurrency and Computation: Practice and Experience*, vol. 27, no. 16, pp. 4091–4106, 2015, cpe.3549, ISSN: 1532-0634.

Theses

1. It becomes apparent that none of the established Industrial Ethernet systems meets all requirements concerning high realiability, scalability, flexibility in terms of self-configuration and cost-effective standard Ethernet hardware. No currently established solution achieves isochronous real-time data transmission without special hardware and single point of failure at the same time.
2. A fully decentralized network can enable hard real-time operations without single point of failure. Such a solution simplifies administrative scalability and usability in adding and removing further instances as no managing central instance is necessary. Every node in such a network is able to act autonomously on its given data in a deterministic way to enable the hard real-time behavior.
3. A comprehensive modeling approach like joint routing and scheduling is the method of choice for routing and scheduling of time-triggered networks. Fixed routing approaches either have drawbacks with respect to schedulability or latencies, while joint routing and scheduling yields good results with respect to both of these aspects. A scheduling approach with fixed load balanced routing shows higher flow latencies and a scheduling approach with fixed shortest path routing cannot schedule as many flows as joint routing and scheduling.
4. However, if the algorithms often need to react to dynamism in the network at runtime even solving medium-size scheduling and routing problem instances becomes an issue in terms of computation times. Approximation algorithms with polynomial runtime can help to reduce computation times but are currently only able to reserve routes and have to be extended by the ability to compute schedules.
5. Wireless mesh networks and applications executed in such networks benefit from cross-layer cooperation in terms of performance and robustness. These optimizations should, according to a cross-layer principle, take into account the behavior of applications and services of the intelligent environment and configure the physical network accordingly, as well as provide information from the lower network layers to higher layers in order to adapt the communication behavior to the given network structure and quality.
6. Distributed applications in dynamic opportunistic networks with highly dynamic participants have been shown to be able to be accurate and scalable. However, the dynamism due to mobility imposes tight constraints on the convergence speed of the algorithm. It is questionable whether an approach different from broadcasting is feasible to exchange information between mobile nodes in highly dynamic scenarios.
7. If the node density in an opportunistic network is sufficiently high, the accuracy of estimates for distributed voting and crowd counting results approaches 100% compared to the real value.

Teaching at the University of Rostock

At the time of writing this habilitation thesis, the author had taught 9 different courses; including three lectures. Only courses are listed, which have been taught after the completion of the author's PhD thesis in October 2012.

Time period	**Topic**	**Type of course**
2017-2018	Selected topics in VLSI design	Exercise
2013-2015, 2017-2018	Digital systems	Exercise
2014, 2016-2018	C++/GUI	Lecture
2015	Distributed embedded systems	Lecture
2014	Technical basics of computer communication	Lecture
2012-2014	Basics of electrical engineering	Practical course
2012-2013	Project-oriented software development for engineers	Exercise
2007-2013	Applied VLSI design	Exercise and practical course
2007-2014	Basics of VLSI technology/Highly integrated systems	Exercise

Supervised Student Works

At the time of writing this habilitation thesis, the author had supervised 35 student works, of which four were awarded, and several students jobs, which are not listed here. Behind the topic of the student work is indicated in brackets, if the respective work was awarded a prize. Only student works are listed, which have been supervised after the completion of the author's PhD thesis in October 2012.

Master Theses

Year	**Topic**
2018	Development of a Localization System for the Material Handling Domain
2018	Development of an OPC UA Framework for the Material Handling Domain
2018	Entwicklung und Bewertung einer verteilten Content-Caching-Lösung für IEEE-802.11s-WLAN-Mesh-Netzwerke
2017	Dezentrale Netzstabilisierung auf Basis der Detektion von Netzfrequenzschwankungen
2015	Entwicklung und Evaluierung eines SDN-gestützten echtzeitfähigen Gerätenetzwerkes *(Springer BestMasters)*
2015	P2P-basierte Verteilung von Ressourcen und Managementfunktionen in IEEE 802.11s WLAN-Mesh-Netzwerken
2014	Integration von Dienstgüteparametern aus physischen WLAN-Mesh-Netzwerken in logische P2P-Netzwerke
2013	Entwurf eines echtzeitfähigen Peer-to-Peer-Clients *(Ludwig-Bölkow-Nachwuchspreis)*
2013	Evaluierung und Optimierung der WS-Discovery für hochskalierende Netzwerke
2013	Konzipierung und Implementierung eines Hardware-basierten EXI-Parsers für ein echtzeitfähiges Web Service-Profil
2013	Konzeption und Entwicklung einer Konfigurations-, Management- und Überwachungs-Lösung für vermaschte, drahtlose Netzwerke basierend auf IEEE 802.11s *(Dr.-Werner-Petersen-Preis der Technik)*

Bachelor Theses

Year	**Topic**
2015	Evaluierung von Cross-Layer-Verfahren für Überlast- und Flusskontrolle in 802.11s-WLAN-Mesh-Netzwerken
2014	Untersuchung der Skalierbarkeit eines P2P-basierten DNS
2014	Konzipierung und Implementierung einer Cloud-gestützten Fernsteuerung von Smart Home-Funktionen mittels OpenStack
2014	Performance-Analyse eines dynamischen Suchtoleranzalgorithmus für das Kad-Netzwerk
2014	Untersuchungen der Skalierbarkeit eines P2P-basierten verteilten Rechensystems
2013	Umsetzung kryptografischer Algorithmen in VHDL
2013	Umsetzung eines dezentralen Synchronisations- und arbitrierten Medienzugriffsmechanismus für das Kad-Netzwerk
2013	Umsetzung eines echtzeitfähigen Datenaustausches mit hoher Ausfallsicherheit in Kad-Netzwerken *(Dr.-Werner-Petersen-Preis der Technik)*
2013	Integration von WS-Eventing in eine dezentrale P2P-Umgebung
2013	Konzipierung und Implementierung eines DHT-basierten Verfahrens für Geräte- und Service-Discovery

Project Theses or comparable

Year	**Topic**
2018	Optimization of MQTT-SN for the Industrial Internet of Things
2018	Simulation Model of a real-time compliant Ethernet Switch
2018	Development of a simulation model of gPTP in OMNeT++
2018	Development of an MQTT-SN simulation model in OMNeT++ and evaluation regarding the Industrial IoT
2017	Berücksichtigung von physikalischer Netzwerk-Nähe im DHT-Protokoll Kademlia
2017	Umsetzung der TSN-Standards Synchronisierung und Frame Preemption in Omnet++
2014	Nutzung von SDN in der MiniNet-Umgebung zur Optimierung einer Beispielapplikation
2013	Entwurf und Evaluation eines Kad-basierten P2P-Systems für Android-Plattformen

Literature Reviews

Year	**Topic**
2014	Gegenüberstellung der Überlast- und Flusskontrollverfahren auf Sicherungs-, Vermittlungs- und Transportschicht
2014	Stand der Technik und Simulationsmöglichkeiten von Peer-to-Peer-basiertem IPTV
2013	Echtzeitdatenübertragung in Automatisierungsumgebungen
2013	Untersuchung von P2P-basierten verteilten Rechensystemen
2013	Untersuchung der Einflüsse von dynamisch regelbaren Suchtoleranzen in Distributed Hash Table-basierten P2P-Netzwerken
2013	Untersuchung von Möglichkeiten zur Cloud-gestützten Fernsteuerung von Smart Home-Funktionen

Statement of Authorship

I declare that this thesis

Network and Application Design for Reliable Distributed Systems

was composed independently and that it does not incorporate, without any acknowledgement, any material previously submitted for a degree at any university. To the best of my knowledge, this habiliation thesis does not contain any materials previously published or written by any other person except where due reference is made.

Rostock, June 1, 2019