Mälardalen University Licentiate Thesis 266

Improving Development of Communication Software in Industrial Control Systems using Simulation

Niclas Ericsson

MÄLARDALEN UNIVERSITY
SWEDEN

Mälardalen University Press Licentiate Theses
No. 266

IMPROVING DEVELOPMENT OF COMMUNICATION SOFTWARE IN INDUSTRIAL CONTROL SYSTEMS USING SIMULATION

Niclas Ericsson

2017

School of Innovation, Design and Engineering

ISBN 978-91-7485-360-5
ISSN 1651-9256
Printed by E-Print AB, Stockholm, Sweden

Abstract

In the industrial domain, customers expect a product longevity of 10-20 years, with high reliability and availability. Since industrial distributed control systems often are safety critical, aspects such as determinism, low latency and jitter are crucial. More and more industrial systems are becoming connected to the Internet, since end customers are requiring e.g. business intelligence and diagnostic information, anywhere at any time. Industrial systems that traditionally have been isolated are now facing entirely new challenges that will require new competences and ways of working. Introducing a new type of network in the industrial domain is a big investment, with high risks, often lacking known best practices. Time to market with sufficient quality is of high importance. A lot of time is spent on isolated activates, such as, simulations, updating tools, collecting requirements, design, coding, debugging, documentation, creating testbeds, validation and reviews. Therefore, there is a need to improve the efficiency when moving between the research and development phases for several reasons, e.g., integrate innovative research findings into industrial systems, shorten time to market, and improve product quality. This thesis focuses on improving efficiency during research and development of communication software. First, network evaluation methods are studied, and key industrial challenges are identified. For example, despite a huge research effort on network simulators and virtualization, there are still challenges that need to be addressed, in order for increased industrial benefits. Secondly, this thesis proposes a flexible communication stack design that supports different run-time behaviors, from real-time operating system to bare-metal systems without an operating system, and different types of communication protocols, from real-time to non-real-time. Finally, this thesis proposes a set of key features from network simulators, that are implemented and used as a case study in a research project. These contributions lead to simplification and increased automation, hence reducing the amount of manual work during research and development.

Sammanfattning

Inom industrin förväntar sig slutkunderna att produkterna har en livslängd på 10–20 år med hög tillförlitlighet och tillgänglighet. De flesta industriella styrsystem automatiserar säkerhetskritiska processer. Detta gör att aspekter som förutbestämt beteende med små fördröjningar och variationer är avgörande för att skydda person, miljö och egendom. Trender som den nya digitaliseringen, sakernas internet, molnet, 5G, maskininlärning och artificiell intelligens, bidrar till att antalet styrsystem som ansluts till Internet ökar. Ökningen beror mycket på att slutkunder börjar förvänta sig nya tjänster, samt tillgång till affärs- och diagnostikinformation även utanför arbetsplatserna. Att introducera nya kommunikationslösningar inom industrin är ofta en stor investering. Riskerna är oftast höga samtidigt som befintliga kommunikationsteknologier och protokoll behöver stödjas även i framtiden. Att få ut produkterna snabbt på marknaden med bibehållen kvalitet är av stor betydelse. Under forsknings- och utvecklingsarbete spenderas mycket tid på isolerade aktiviteter som till exempel simuleringar, uppdateringar av verktyg, insamling av krav, design, programmering, felsökning, dokumentation, testning och granskningar. Att snabbt byta mellan dessa aktiviteter är ofta inte helt enkelt, på grund av att olika metoder och verktyg inte automatiskt kan utbyta eller överföra information. Fokus i denna avhandling är att förbättra effektiviteten vid forskning och utveckling av kommunikationsmjukvara. Effektiviteten är viktig av flera skäl, till exempel att snabbt integrera nya och innovativa forskningsresultat, snabbare nå marknaden med produkter och förbättra produktkvalitet. Initialt studeras olika metoder för att utvärdera kommunikationsfunktionalitet. Metodernas användbarhet kartläggs i förhållande till aktiviteter under forsknings- och utvecklingsarbete, samt viktiga industriella utmaningar identifieras. Trots stora forskningsinsatser inom nätverkssimulatorer, emulatorer och virtualisering, så finns det fortfarande utmaningar kvar för ökad användbarhet och nytta inom industrin. Vidare föreslås en flexibel kommunikationsstackdesign som stöder

olika typer av egenskaper och implementationer, från realtidsoperativsystem till enheter helt utan operativsystem, samt olika typer av kommunikationsprotokoll, från realtid till icke-realtid. Slutligen föreslås en reducerad uppsättning nyckelfunktioner till nätverkssimulatorer, vilka har implementerats och använts i en fallstudie i ett forskningsprojekt. Dessa bidrag tillsammans medför en förenkling och ökad automatisering vilket gör att mängden manuellt arbete minskar under forsknings- och utvecklingsarbete.

List of Figures

List of Tables

To my family

Acknowledgments

Many thanks to my supervisors, Mats Björkman (MDH), Johan Åkerberg (ABB Corporate Research) and Tomas Lennvall (RISE SICS Västerås), who tricked me into taking this journey as an Industrial PhD student. All the supporting discussions and guidance have been crucial for this licentiate thesis.
I also have to thank my manager, Helena Jerregård (RISE SICS Västerås), and my previous manager Linus Thrybom (ABB Corporate Research), that made it possible for me to start as an industrial PhD student.
Furthermore, thanks to all my colleagues at RISE SICS Västerås, and all my former colleagues at ABB Corporate Research, for all discussions, reviewing and support. I especially would like to thank Stig Larsson, Ali Balador, Kristian Sandström, Markus Bohlin, and my fellow PhD students at SICS, Sahar Tahvili and Daniel Flemström, as well as my former colleagues Krister Landernäs, Ewa Hansen.
The work presented in this licentiate thesis has been funded by RISE SICS Västerås, ABB Corporate Research, Vinnova through the IoTSP project and the Swedish Knowledge Foundation (KK stiftelsen) through the ITS ESS-H program at Mälardalen University.
I dedicate this licentiate thesis to my wife Malin and my son Noel.

Niclas Ericsson
Västerås, December 2017

List of Publications

Papers Included in the Licentiate Thesis [1] [2]

Paper A *Challenges from research to deployment of Industrial Distributed Control Systems*. Niclas Ericsson, Tomas Lennvall, Johan Åkerberg, Mats Björkman. The 14th IEEE International Conference on Industrial Informatics, Poitiers, France, July, 2016.

Paper B *A Flexible Communication Stack Design for Time Sensitive Embedded Systems*. Niclas Ericsson, Tomas Lennvall, Johan Åkerberg, Mats Björkman. The 18th Annual International Conference on Industrial Technology, Toronto, Canada, March, 2017.

Paper C *Custom simulation of Industrial Wireless Sensor and Actuator Network for improved efficiency during Research and Development*. Niclas Ericsson, Tomas Lennvall, Johan Åkerberg, Mats Björkman. The 22nd IEEE International Conference on Emerging Technologies and Factory Automation, Limassol, Cyprus, September, 2017.

[1] A licentiate degree is a Swedish graduate degree halfway between M.Sc. and Ph.D.
[2] The included articles have been reformatted to comply with the licentiate layout.

Papers Not Included in the Licentiate Thesis

1. *Communication Middleware Technologies for Industrial Distributed Control Systems: A Literature Review*. Ali Balador, Niclas Ericsson, Zeinab Bakhshi. The 22nd IEEE International Conference on Emerging Technologies and Factory Automation, Limassol, Cyprus, September, 2017.

Contents

I

Thesis

Chapter 1

Introduction

The amount of communication in the industrial domain is growing. This is accelerated by the digitalization trend with buzz words like IoT, 5G and Cloud, that are introducing new communication technologies and protocols. Many industrial Distributed Control Systems (DCS), are safety critical with end customers that are used to a product longevity of 10-20 years, with an availability of up to 99.9999% [1]. Combining these traditional systems with the recent trends are challenging. The devices and run time contexts of embedded systems in industrial networks vary, from sensors and actuators with microcontrollers without any operating system (OS), to controllers with multi-core processors that run real-time operating systems (RTOS). The communication also varies, from diagnostic data without any hard time constraints i.e., best effort, to safety critical real-time data that must be delivered on time. Introducing a new type of network in the industrial domain is a big investment, often lacking known best practices. The entire distributed control system must work deterministically in order to provide the desired function, thereby affecting all R&D phases and all involved components. Development cost, time to market and product quality are of high importance. The longevity, safety and security requirements have a high impact on the entire product lifecycle, by increased development effort and risks. For new products or in case of a product upgrade, it is not just the system requirements that must be fulfilled and validated, other task also needs to be carried out, such as, hardware certifications, safety and security assessments. Therefore, detecting and resolving problems as early as possible is important, problems found late in development or in deployed systems, tend to be costly to fix [2]. The solution and choice of hardware and software com-

ponents depend on the overall system requirements. Note that while adopting to new end customer demands, industrial suppliers must still support a legacy of field buses and protocols. Hence the need for compatibility, changeability and reuse. Therefore, in order to address the increased complexity, there is a need to improve the efficiency when developing communication software, by automating, simplifying and removing unnecessary work in and between the research and development (R&D) phases. Network evaluation in industry is commonly done using testbeds, since the setup can be similar to the real deployed system. However, as the networks grow and become more complex, using, maintaining and extending a testbed becomes more and more costly, due to the increase in cost for hardware, labor and troubleshooting. Therefore, other network evaluation methods like simulators and virtualization in combination with testbeds are envisioned to further streamline the workflow, enabling a faster pace in adopting new and innovative research findings, shorten time to market and improve product quality.

The focus of this thesis is to address the challenges of R&D efficiency of communication software by using simulation at various abstraction levels throughout the development phases. By developing a new type of simulator, which is simple to implement and use, error prone manual labor is reduced through increased automation and adjustable levels of abstraction, enabling easy adaption to the requirements of each development phase.

1.1 Research Problem

Requirements such as safety, security, availability and longevity in harsh industrial environments affect the entire industrial product lifecycle. Maintaining and evolving industrial DCS systems, while introducing new research findings is challenging for many reasons, like lack of tool integration, manual labor introducing errors, complex tools requiring expert knowledge to use, setup and maintain. Quite often are technology from the consumer domain, such as Ethernet, WiFi or Bluetooth adapted and used in the industrial domain. However, in order to meet the required availability and deterministic system behavior, modifications are needed. For example, network topologies are designed to support redundancy, e.g., ring topology for wired and mesh topology for wireless communication. The required deterministic system behavior affects the software in the devices and the communication between devices, hence the common use of RTOS and real-time communication protocols within industry. Network evaluations in industry are commonly conducted on testbeds in vari-

ous forms, from a couple of devices to a more representative testbed. Testbeds are however seldom set up for the maximum configuration, due to the cost of both hardware and labor. Notable is that creating an advanced network topology like a wireless mesh may be challenging in itself, even when everything works. Hence recreating phenomena seen earlier during e.g., evaluations or in deployed systems may be infeasible in a testbed. Other network evaluation methods such as simulation and virtualization are providing scalability, repeatability, and control of network aspects such as, topology, disturbances and signal strength. However, even though there has been a significant research effort on various network evaluation methods, all with different abstraction of reality, none of them fit all needs through all the Research and Development (R&D) phases. This highlights the need to change network evaluation methods when moving from early research and development to the final product phase in a simple, highly automated way. However, switching back and forth between different network evaluation methods can be challenging. For example, code used to verify an algorithm in the early research phase can seldom be reused in later phases of development of industrial products due to different aspects like, programming language, and run-time context. Advanced tools may be costly to buy, labor intensive to setup and maintain, and complex to use, while still not solving the overall problem. Therefore, a new approach is needed that streamline the workflow by automating, simplifying and reducing unnecessary work throughout the R&D phases.

1.2 Research Questions

The overall goal of this thesis is to develop a new approach to streamline the workflow from research to deployment of industrial distributed control systems. The focus is on simplicity and ease of use, higher level of automation to reduce manual and error prone tasks, while supporting different abstraction levels dictated by the different R&D phases. A further goal is to simplify the workflow when problems occur late in the development, requiring rapid iterations to earlier phases. The following research questions that are a subset of the overall goal will be addressed:

RQ1 How can code reuse between simulations and deployment on time sensitive target systems be improved?

RQ2 What features in simulators can make the R&D workflow of communication software more efficient?

RQ3 How can a communication stack design and implementation efficiently support different run-time contexts (from bare metal super loop, to multi-threaded RTOS), and different types of communication protocols, from non-real-time (best effort), to real-time?

1.3 Hypotheses

Network simulation can be used with real target system code to speed up going from research theories to deployment of time sensitive industrial communication stacks and protocols. The real source code can be run in a network simulator with similar behavior as industrial real-time devices regarding timing and run time system primitives, (e.g. threads, events, semaphores). When it comes to developing software of the upper layers (above the physical layer), a small subset of network simulator features can be sufficient to improve R&D efficiency. In order to improve changeability and reuse, a communication stack design and implementation can be run-time agnostic and support being configured for time sensitive (real-time) or best effort (non-real-time) communication.

1.4 Research Approach

Research and development of industrial distributed control systems is hard, due to many different reasons. Requirements for safety, longevity, and legacy in combination with challenges related to the fact that the systems are time sensitive and distributed, have an impact on the industrial products entire lifecycle. In order to understand the research area and verify the problem, a state of the art review on network evaluation methods was conducted. The resulting analysis shows that when it comes to network evaluation methods and tools, no solution fits all needs for industrial DCS throughout all activities in the different R&D phases. Therefore, the approach is to identify specific real world needs and challenges like debugging, and apply an engineering approach rather than a theoretical approach, in order to capitalize on and improve existing solutions related to network evaluation.

The research methodology is of an applied nature. Primarily by observing existing solutions, proposing an improvement, building a proof of concept, measuring and analyzing. Proof of concept implementations, applied as case studies in existing R&D projects, have been used to validate the solutions. The resulting data is then collected through interviews with project members, i.e.

the conclusions and suggestions are based on the project team members' collected experience. The goal of this thesis is to propose solutions that automate, simplify or reduce unnecessary work throughout the R&D phases.

1.5 Thesis Contribution

The main contributions of this thesis are:

1. Identification of key challenges posed by DCS requirements such as compatibility, changeability, reuse, granularity, and debugging, related to different network evaluation methods in relation to R&D phases.

2. A proof of concept implementation, showing that a flexible stack design can support real-time communication with an RTOS or non-real-time communication with or without an OS, thereby improving reuse and changeability, between different types of communication and run-time contexts.

3. A new approach to allow easy switching back and forth between testbed and simulator throughout the R&D phases, with a recommended subset of network simulator features, and a proof of concept implementation used in an industrial case study.

The combination of these contributions shows that it is feasible to streamline the workflow from research to validation, by automating, simplifying and removing unnecessary manual work in and between the R&D phases.

1.6 Thesis Outline

This thesis is a collection of papers, consisting of two parts: Part I starts with Chapter 1 that contains research problem, questions, hypotheses, approach and an overview of the contributions, followed by Chapter 2 containing background information and definitions. Chapter 3 presents related work. Chapter 4 presents the thesis contributions and an overview of included papers. Chapter 5 contains conclusions, reflections and future work. Part II, Chapter 6 - 8, includes three peer-review scientific papers that are published and presented in international conferences.

Chapter 2

Background

There are several recent trends related to digitalization, like Internet of Things (IoT), Edged Cloud, 5G, and Cyber Physical Systems (CPS), that are pushing for increased communication and connectivity to the Internet. These trends are affecting industrial systems, since more and more systems are becoming connected to the Internet, and end customers are requiring e.g., business intelligence, and diagnostic information, anywhere at any time. Figure 2.1 shows a typical automation system sometimes referred to as the industrial automation pyramid, in combination with IoT Edge and Cloud technologies. The first level at the bottom of the pyramid contains sensors and actuators distributed throughout the process, enabling production equipment and machines to be controlled. The communication to the next level is via field buses. The second level contains controllers and realizes automatic control functionality. Process data from the sensors are inputs to the control loops and the outputs are sent to actuators as set points. The third level provides supervisory control of the production sequence, communication with the PLCs via a control network and the next level via a client/server network. The fourth level, production/batch control, handles the work flows and communicates with the next level via a plant/corporate network. The fifth level at the top of the pyramid is the enterprise or management level, here are all business-related systems and activities handled, like communication with different plants, suppliers and customers. IoT, Edge and Cloud technologies are likely to be inserted at different levels of the automation pyramid, depending on business and use case. Cloud services such as big data with machine learning may very well be the key enabler to realize new or improved services, like predictive maintenance or process

optimization.

The communication throughout the pyramid covers heterogeneous networks with various characteristics and requirements, including many different communication technologies and protocols. The Cloud, Enterprise and Production/Batch Control level are usually handled by standard information technology (IT). The Supervisory Control level is a mix of IT to interface upwards, and operational technology (OT) to interface the physical equipment downwards, in the pyramid. Communication to the Edge where Cloud technology is used locally, is also likely a mix of IT and OT, depending on the level in the pyramid to interface. The Automatic Control and Sensor and Actuator level are handled by standardized OT. On these levels, often the whole system is time sensitive, thereby requiring real-time behavior of the devices and communication network between devices.

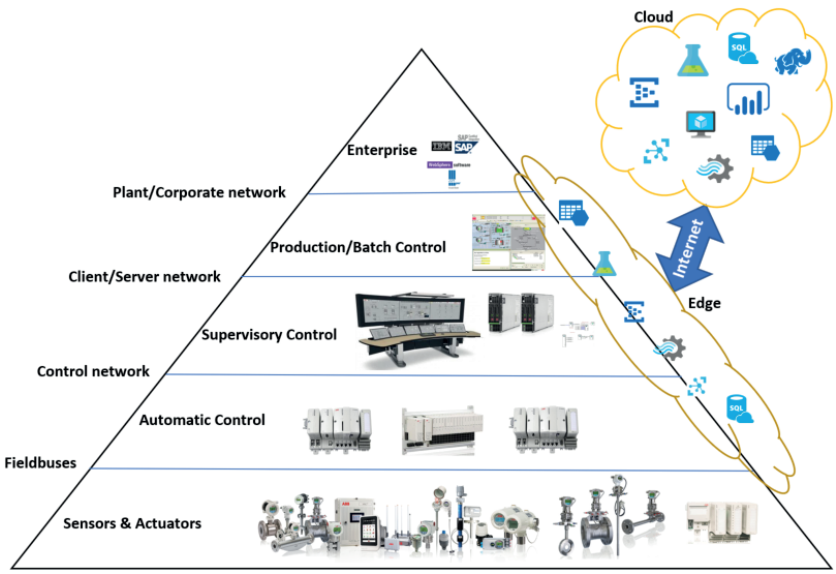

Figure 2.1: Industrial Automation Pyramid.

IoT, Edge and Cloud, lead to an increase in the need for IT and OT convergence. Development and product cycles are short in the IT domain. On the other hand, development in the OT domain often take years, due to the increased amount of work caused by the industrial requirements, such as, safety,

longevity and the harsh environment. The longevity of industrial systems, adds additional effort to the R&D process, since the system needs to evolve into the next generation, rather than be replaced, in order to support both legacy and new functionality [3]. However, Internet connectivity will require more frequent updates of the products in the OT domain, due to cyber security vulnerabilities. New cyber security threats and counter measures appear all the time, therefore making security more of a continuous process than a state as in safety.

2.1 Research and Development phases

Research and development goes through several different phases, often described as different development models, e.g. Agile, V-model or the waterfall [4] shown in Figure 2.2. They all describe different phases and activities, and several of the phases are iterative. This means that you can for example go from the Maintenance to the Research phase in order to evaluate new concepts, or from the Verification to the Requirements phase if incomplete or incorrect requirements are detected.

Figure 2.2: Research and Development phases.

R&D of communication software use different network evaluation methods to validate different aspects such as performance and intended behavior. When it comes to network evaluation, the needs change depending on R&D phase and activity. Below is a high-level overview of the different aspects related to R&D of communication software and network evaluation.

Research phase often starts with theories, using for example, network simulators to develop and validate an algorithm. Even though simulation is an abstraction of reality and contains simplifications it provides important aspects, e.g., network scalability, repeatability, a high degree of control, all at a low cost.

Requirement phase is usually the start of product development, focusing on specifying the functionality of the system and the different parts.

Design phase follows with activities such as analysis and benchmarking for

selecting hardware and software components and tools. A lot of effort goes in to creating a system architecture, that fulfills the requirements.

Implementation phase is when the system architecture is realized. For software developers it involves for example, coding and debugging. Programmers creating new communication stacks or protocols commonly utilize loopback interfaces, in order to debug and validate that everything works under perfect conditions. When the target is an embedded system, a small testbed is often used by each developer. However, debugging on a network testbed may be challenging, since it is hard to inspect the states of multiple devices simultaneously.

Verification phase is when the full system is integrated and validated, often by creating a large testbed, which is costly and time consuming to set up and maintain. In addition to the challenge of debugging on a testbed, it is hard to manually perform repeatable network evaluations, such as packet loss, interference and topology changes. However, since a testbed is the closest evaluation method compared to a deployed system, it is a necessary step before piloting in the field.

Maintenance phase focus on patches and upgrades. Due to the longevity of industrial products (10-20 years), hardware and software components and tools may need to be upgraded or replaced, testbeds may have been decommissioned and may be hard to recreate.

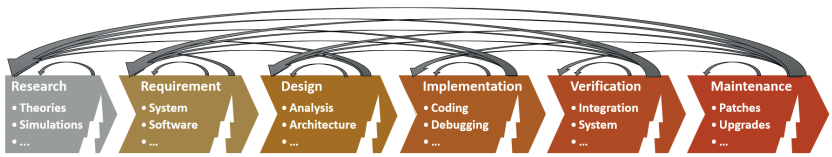

Figure 2.3: Research and Development phases with iterations and gaps.

As mentioned earlier, aspects such as, new customer requirements and bug fixes may cause an iteration, and thereby forcing a product to go through all of the R&D phases again, and moving between and within different phases may be challenging. For example, due to various data formats for different tools, thereby making it hard to achieve traceability from requirement to verification for safety assessment. Figure 2.3 shows the possible iterations between R&D phases, and the gaps between the different R&D phases and within phases.

2.2 Network Evaluation Methods

The need for network evaluation differs and depend on a lot of different aspects from network topology, communication medium characteristics to activity and R&D phase. There are many definitions that are used in various contexts. For example, the use of words like simulation, emulation and virtualization may have different meaning to different individuals and domains. Network evaluation methods are often described as different techniques [5] or categories [6] and different levels of abstraction relative to the real application. From theoretical analysis, as the highest abstraction, followed by simulation, emulation, virtualization, real testbed and real deployment.

2.2.1 Analysis

Theoretical Analysis or analytical models, uses a set of mathematical equations to create a model of a system, that can give quick insights at a low cost. Although theoretical analysis simplifies the modeling procedure, analysis of advanced networks, like wireless mesh networks is very difficult and useful mathematical tools are lacking, due to the complexity of mathematical constructs for realistic considerations [6].

2.2.2 Simulations

Simulations provide a controllable and repeatable environment for network studies. Studies are flexible at low cost, since many experiments can be conducted on a single computer. Despite the environment being an abstraction of reality, containing simplifications that may lead to unrealistic results compared to real-world measurements, simulation enables the study of large scale and complex network scenarios that may not be possible to perform in a real testbed.

There are two types of simulations that are widely used [5]. Trace driven simulation uses a trace of a time ordered history from a real system as input to the simulation process. Discrete event simulation model operations as a sequence of events. An event occurs at a specific time and all events are run in sequence, thereby enabling skipping idle time between operations.

Simulation can be performed on different levels [7], such as application, operating system and hardware level. The level affects the execution efficiency of the simulator as well as software development. These three different abstraction levels can also be described differently [8, 9]. Below are the definitions

used in this thesis.

Application level simulators work on the networking level, and do not model hardware or platform specific APIs. The source code is only intended for the simulator itself, which makes the simulations fast and useful for studying high level algorithms.

Hardware level simulators or node emulators or instruction set simulators, simulate a specific platform at the hardware level. This enables running unmodified binaries compiled for the target system, but at the cost of longer simulation times and higher code complexity in the simulator.

Operating System level simulators enable running source code compiled to native code for the simulation host, by providing identical APIs for the OS and HW peripherals. This platform approach is more efficient than hardware level simulation, but in addition to not using all the target system code, the binaries must be compiled with another toolchain.

2.2.3 Emulation

Emulation is a hybrid environment consisting of a simulated environment and real hardware with real network layers. Emulation has the advantage over simulation that it can validate against real traffic. Compared to real testbeds, emulation provides some of the advantages of simulation, such as, repeatability, control. However, scalability is limited and using emulation for larger networks may not be feasible, since, time synchronization is especially challenging for hybrid solutions [10].

2.2.4 Virtualization

Virtualization environments simulate the hardware resources, such as CPU, network, storage and I/O. Software development can be performed on real machines with a real OS, and tested on the virtual network of virtual machines. CPU virtualization include application and processing virtualization [11], the later covers a range of technologies allowing one computer appearing as many or many computers as one.

Application virtualization or software virtualization, enables applications to be encapsulated and run in an artificial environment, thereby breaking dependencies between applications and allowing instant application activation, deactivation and reset to default settings.

Processing virtualization or system virtualization provides a complete virtual hardware representation, enabling multiple and unmodified operating systems (OS) to be executed on the same host machine. A hypervisor, also referred to as monitor, is responsible for managing the physical hardware and each virtual machine. Hypervisors are generally described as, bare metal hypervisors that run directly on the system hardware, or hosted hypervisor that run on a host OS that provides virtual resources. A hypervisor often supports different abstraction levels of virtualization of the hardware. Full virtualization refers to when the exposed virtual hardware is identical to the underlying machine, thereby supporting unmodified guest OS, such as Windows or Linux. Paravirtualization is another abstraction were the virtualization is similar but not identical to the underlying machine. By modifications to the guest OS, better performance can be gained, since some supervisor instructions must be handled by the hypervisor for correct virtualization [12]. A hypervisor can take advantage of hardware assisted virtualization support that are available in different processor architectures, in order to improve performance of full virtualization.

Operating System level virtualization also known as containers, is a type of system virtualization that sits on top of a physical server and its host OS. The guest and the host OS are the same, since the host OS kernel is shared and used to implement the guest environment.

2.2.5 Testbeds

Real testbeds is a network of real target systems that can be very similar to the real environment of where the devices will be deployed. However, studies often lack in control and repeatability, especially in case of distributed wireless networks. Testbed are restricted in size as well as in complexity, due to high expenses depending on both hardware costs and labor intensity, however testbeds bridge the gap to deployment, and are a necessary step.

2.2.6 Piloting and Deployment

Real world deployment is the final network evaluation, often one or a few installations are selected for piloting, a final experiment or study before market launch. Since the deployment is performed in the targeted environment of the product, no inaccurate or incorrect assumptions are made. However, control and repeatability is lacking and for industrial products, finding and getting access to the most challenging environmental conditions may be hard.

2.3 Summary

Recent trends are pushing for increased functionality and connectivity, thereby making already advanced industrial system, even more complex and challenging to develop and maintain. As mentioned earlier, there are many different development models with different strengths and weaknesses that affect the software development life cycle [13]. The work in this thesis is not directly tied to any specific development model. The waterfall was selected as an example, to emphasize the software development complexity, and that R&D is not as straight forward as it seems. To my knowledge, no network evaluation method or tool are able to support all industrial aspects with heterogeneous networks where determinism and timing are crucial, throughout all the R&D phases. Notable is that a lot of challenges related to software engineering have been research topics for several decades and still is, such as reuse [14].

The focus of this thesis is improving efficiency during R&D of communication software, by using a combination of different network evaluation methods, specifically network simulation as a complement to testbeds in order to automate, simplify and remove unnecessary work.

Chapter 3

Related Work

There is a lot of research related to improving the efficiency during R&D. The amount of research on software engineering is impressive and highly relevant, since the research field addresses every stage of the development process, looking at aspects such as, development models [15], cost estimations [16] and software reuse [14]. The research processes of software engineering vary and may not be explicitly identified or explained [17], especially to other scientists, even in computer science. The work in this thesis is in the research fields of distributed systems and software engineering, focusing on communication software in industrial DCS.

As mentioned earlier there are several different methods for evaluating networks, such as analysis, simulation, emulation, virtualization and testbeds. However, the different levels of abstraction related to network evaluation methods make them more or less useful depending on the R&D phase and activity. In contrast to the work presented in this thesis, most current methods focus on one or a few specific R&D phases. One of the more common mathematical tools is queuing theory [6], and dates back to the late seventies, e.g., queuing network analysis of computer communication networks [18]. For network evaluations using theoretical analysis, queuing theory is one common mathematical tool [6]. However, as networks become more advanced and complex, like wireless mesh networks, theoretical analysis becomes very difficult [6], due to that analytical models require simplifications and are difficult or impossible to deploy [19].

Network simulation provides a controlled and repeatable evaluation environment. A lot of research has focused on simulation speed, e.g. by paralleliz-

ing and distributing the simulation work [20, 21]. One of the early simulators that highlights that wired and wireless networks have fundamental differences is GloMoSim [22]. For example, for wireless media the signal interference and attenuation are more complicated, hence simulations is more computation intensive, than for wired. Since wireless communication is more challenging and less mature than wired communication, researchers have changed focus to simulation of wireless sensor networks (WSN). Imran et. al claim [23] that simulation is the most popular, effective and feasible approach to design, develop and test network protocols for WSNs. Fall claims [24] that simulation and testbed construction represent the two most important methodologies available for design and evaluation of network protocols, and that hybrids or simulation and testbed can be combined into a network emulator. However, the time synchronization is especially challenging for hybrid solutions [10], thereby limiting scalability of the simulated network [5]. The accuracy is affected by how detailed the system is modeled and determinism of the computing of the model [25]. There are also research approaches that combine node virtualization with network emulation [26, 27, 28]. The time sensitivity of industrial DCS is most likely to further limit the scalability, or even make the use of emulation and mixes with node virtualization infeasible, depending on the application requirements.

Network virtualization allows multiple virtual networks to share one physical network, thereby achieving more flexibility, diversity, security and manageability [29]. There are also network simulator/emulators that utilize lightweight virtualization (i.e. containers) and are able to create a virtual testbed, running real Linux applications. For example, Mininet [30] and CORE [31] that use paravirtualization to only make the process and TCP/IP network stack virtual. There are also research efforts for virtualization of wireless, with other challenges, such as isolation, management and security [32]. A recent survey envisions future WSN to support multiple applications simultaneously [33]. The survey goes through node-level and network-level virtualization solutions, as well as hybrid solutions of node and network virtualization. From an industrial and software development perspective the RTOS called RIOT [34] is interesting, since software development for RIOT can be done natively in a virtual testbed on Linux or Mac OS [35]. However, RIOT development does not support Windows that is commonly used and required by a majority of industrial companies, and in addition time synchronization between the virtual nodes is unclear.

There are several public testbeds, like PlanetLab [36], CitySense [37], and Sensei-UU [38]. However, developing and maintaining such platforms are very

difficult, which is indicated by the fact that several testbeds have been decommissioned [39], e.g., MoteLab [40] and TrueMobile [41]. From an industrial perspective, the usefulness of public testbeds is questionable, due to reasons like, protecting intellectual property, and whether the hardware and software platforms match the industrial needs.

Different network evaluation methods, simulation levels and types of virtualization fulfill specific need in R&D, however none of them fit all needs in all R&D phases. Many of the simulators mitigate this by supporting multiple methods or simulation levels. E.g., OMNeT++ [42], ns-3 [43], and Simics [44] support both network simulation and emulation. Several tools also support multiple simulation levels, e.g., COOJA [7], and Simics [45]. However, due to different abstractions, this often result in that the code cannot be reused between different network evaluation methods and simulation levels.

Simulators that work on the hardware level, also known as instruction set simulators (ISS) or node emulators [44, 46, 47, 48], can provide deterministic timing and support a variety of different hardware architectures, communication stacks and OSes. These tools can run unmodified binaries for different processors such as ARM, x86 and PowerPC, at the cost of longer execution time [7] and that board specific adoptions usually needs to be made for each specific node/device. In addition, hardware level simulators may not be that useful in the research phase, since the hardware platform may not be known or even relevant for early algorithm and protocol evaluations.

Sharif et. al has made an extensive survey on ubiquitous sensor network simulation and emulation environments [5]. After comparing 130 simulators and emulators they concluded that there is no standard simulator or emulator for all ubiquitous sensor network applications. Therefore, recommending one for the industry with heterogeneous networks where determinism and timing are crucial may be even more challenging.

From a reuse of source code between real target systems and network simulators perspective, many simulators support a specific software platform and communication medium, for example, simulators for Wireless Sensor Nodes (WSN), like TOSSIM [49] and COOJA [7]. Both TOSSIM and COOJA are tailor made and is running energy efficient OSes with tasks that are non-preemptive, this may however have a negative impact on latency and real-time performance [50] and therefore not be applicable in industrial systems. Other interesting approaches for reuse of source code is for example, the Linux based Direct Code Execution (DCE) [51] and its ancestor Network Simulation Cradle (NSC) [52]. However, many industrial companies are Windows based, thereby changing to a Linux based simulator on a server or in a virtual machine may be challeng-

ing for many reasons, such as organization policy, performance degradation, and manual labor, hence making the move too time consuming and therefore not worth the effort if the same evaluation can be done more easily on a real testbed. Notable is also that devices in industrial DCS vary, from bare-metal micro-controllers (MCU) without any operating system (OS), to multi-core processors with real-time operating systems (RTOS).

Over the years there have been a lot of research on design patterns in general and work on communication stack design. Communication stacks are often arranged in a layered manner. When it comes to wired communication, there is relatively little details about the design, most likely depending on the maturity of wired communication and that a lot of the effort goes into standardization. A lot of the design or patterns are targeting the upper networking layers (e.g., the C++ libraries ACE, POCO, boost) or one specific solutions (e.g. lwip [53]). Many of the research results on communication stack design patterns are more than 20 years old, and have been implemented in the ADAPTIVE Communication Environment (ACE) [54], such as the architectural pattern Half-Sync/Half-Async [55], Acceptor and Connector [56], Reactor [57], and Active object [58].

For research on wireless communication, there are more details available, for example, the System Architecture Directions for Networked Sensors [59], that has inspired for example, Contiki [60] and TinyOS [61]. Note that both Contiki and TinyOS also have their own customized simulator, COOJA [7] and TOSSIM [49]. Highlighting needs for resource constrained embedded systems and battery powered devices, thereby requiring the scheduler to be power aware. Tasks that are atomic (non-preemptive) are promoted. The later may however not be the primary concern for industrial use, due to a potentially negative impact on latency and real-time performance [50]. In order to archive low latency and jitter, industrial communication needs another communication stack design, for example, the multi-threaded RTOS-based architecture for Industrial Wireless Sensor Network stacks with Multi-Processor support [62].

Chapter 4

Thesis Contributions and Included Papers

The scientific contribution of this thesis is composed of Paper A to C as follows.

Paper A is identifying challenges when going from research to deployment of Industrial Distributed Control Systems (DCS). Since communication is a central part of distributed control systems, network evaluation methods were analyzed and mapped to the R&D phases. The analysis and mapping were based on a literature study and experiences from development of industrial products and platforms. Despite significant research efforts on various network evaluation methods, all with different abstraction of reality, none of them fit all needs, through all the R&D phases. There are still challenges that need to be addressed related to industrial DCS, such as compatibility, changeability, reuse, granularity, and debugging.

Paper B presents a design pattern including a proof of concept implementation and evaluation of a flexible communication stack design. The flexibility improves changeability of the layers in a stack, and how the layers are driven and prioritized. In addition, code reuse between different communication types and run-time contexts are improved, from time sensitive communication with an RTOS to best effort communication with or without an OS.

Paper C proposes a new approach to allow easy switching back and forth between network simulator and testbed, by promoting a customized simulator as a complement to an industrial testbed. A subset of key features from network simulators were recommended and implemented as a proof of concept simulator in a research platform using the stack design presented in Paper B. The

customized simulator was then applied as a case study in a research project, that needed to get from research theories to pilot deployment. The results gathered from the project team, indicate improved efficiency when moving back and forth from the research to the validation phase.

4.1 Paper A

N. Ericsson, T. Lennvall, J. Åkerberg, M. Björkman, *Challenges from research to deployment of Industrial Distributed Control Systems*, The 14th IEEE International Conference on Industrial Informatics, Poitiers, France, July, 2016

Summary

A trend in the industrial domain is that the networks are growing and becoming more complex, this is further accelerated by the digitalization trend. In order to address this, there is a need to improve the efficiency when moving in and between different R&D phases. For example, integrate innovative research findings into industrial systems, shorten time to market, and improve product quality. Despite a huge research effort on e.g., network simulators and emulators, there are still issues that need to be addressed for industrial adoption and benefits. This paper maps various network evaluation methods to different R&D phases, highlighting strengths and weaknesses depending on activity and needs. A set of related key challenges are identified, such as the real-time aspects of industrial DCS, compatibility, changeability, reuse, granularity, and debugging. Resolving issues earlier is the most important improvement. Many challenges are common for networking systems. The main differencing factors for industrial distributed control systems is the longevity, reliability and determinism. Currently no network evaluation method or simulation level outperforms all others throughout the different development phases. Hence there is also a need to improve the transition between network evaluation methods / simulation levels, in order to streamline the workflow. For example, taking findings from the research phase that used application level simulation, to deployable code on industrial testbeds during the development phases.

Method

Analysis based on literature study and experiences from development of industrial products and platforms.

My Contribution

I was the main author and made a state of the art (SOTA) study on network evaluation methods. Based on the SOTA and my own experiences from working as a software developer on communicating industrial products and platforms, and the experiences of the co-authors, I made a gap analysis and mapped the network evaluation methods to the research and development phases. The co-authors provided a lot of guidance, on the work and on writing papers.

Relation to Research Questions

The paper highlights key challenges that need to be addressed for the overall goal to, streamline the workflow from research to deployment of industrial DCS, and aided in identifying the research questions RQ1, RQ2 and RQ3.

4.2 Paper B

N. Ericsson, T. Lennvall, J. Åkerberg, M. Björkman, *A Flexible Communication Stack Design for Time Sensitive Embedded Systems*, The 18th Annual International Conference on Industrial Technology, Toronto, Canada, March, 2017

Summary

Industrial resource constrained embedded systems are facing an increase in code base and functionality, accelerated by trends like IoT, 5G and Cloud that are pushing for connectivity to the Internet. In order to reduce development and maintenance costs, there is a need to increase code reuse between systems with different application requirements. This paper propose a flexible communication stack design, that can be configured for time sensitive communication with a real-time operating system (RTOS), or configured for best effort communication with or without an operating system (OS). The proposed design enables communication layers in stacks to be agnostic of run-time context and unaware of adjacent layers, thereby improving the source code reuse between different systems, as well as reusing specific layers among stacks, e.g, within a device, and in a research or product platform. In addition, changeability is improved, due to the possibility to evaluate different communication stack configurations easily, e.g., changing layers, tuning thread priorities, or specifying run-time

context of a layer or the entire stack. Experiments demonstrate the flexibility and simplicity of the design with different configurations, and round-trip measurements show performance related to run-time context. A system with an OS adds approximately 4.1% overhead compared to a system without an OS, showing that the increased complexity of having no OS may not be worth the increased development effort. To support time sensitive communication, both an RTOS and more than one thread driving the layers is needed. Since each layer may not require its own thread, multiple layers can share thread and queue, thereby improving resource allocation within a communication stack. Measurements show that each thread in average adds an additional overhead of 17%. Although a time sensitive communication stack can reduce the number of threads, the main contribution with this design is that it is run-time agnostic, with latencies that scale as expected with the different run-time contexts and number of threads.

Method

Experiment with proof of concept implementation and analysis of results.

My Contribution

I was the main author, came up with the design, implemented the proof of concept, made the measurements and analysis. The co-authors provided a lot of guidance, for the analysis, how to write a paper, and language formulation.

Relation to Research Questions

This paper addresses RQ3.

4.3 Paper C

N. Ericsson, T. Lennvall, J. Åkerberg, M. Björkman, *Custom simulation of Industrial Wireless Sensor and Actuator Network for improved efficiency during Research and Development*, The 22nd IEEE International Conference on Emerging Technologies and Factory Automation, Limassol, Cyprus, September, 2017

Summary

Recent trends are pushing for an increase in connectivity, but introducing a new type of network in an industrial distributed control system is a big investment with high risks. Time to market with sufficient quality is crucial. For verification and validation of communication software, the most common network evaluation method in industry are real testbeds, mostly since a testbed can be very similar to the deployed system. Testbeds are, however, hard to debug and costly to maintain. Other network evaluation methods, like simulators, have some strengths that testbeds are lacking, like repeatability, control over the network, and lower cost. However, code from simulators can seldom be reused, especially in industrial time-sensitive target systems, due to different abstraction levels, run-time behavior and system timing. This paper presents findings from a case study conducted in a research project that target improved efficiency, getting from research theories, to deployed devices in a homogeneous Industrial Wireless Sensor and Actuator Network (IWSAN). The core of the proposed approach is to adapt the simulator to the reality at hand e.g., platform, time and resources, when there is a need, rather than adjusting the reality to a simulator. The contributions are: improved changeability and code reuse from network simulator to network testbed, and a recommended subset of network simulator features, e.g., control over topology, disturbances and signal strengths, providing tunable granularity and easier debugging in the simulator than the testbed. The selected simulator features are evaluated with a proof of concept implementation that is customized to a research platform. The research platform was used by a team of researchers, working on evaluating new algorithms and developing a complete IWSAN stack for piloting in the field. Results indicate that an advanced and complex network evaluation tool may be overkill, and that ease of use when changing from a simulator to a testbed, and sufficient simulation speed, are more important aspects for improved efficiency.

Method

Experiment with proof of concept implementation applied as a case study in a research project at a company.

My Contribution

I was the main author, made the design and implemented the proof of concept. The co-authors were the users of the simulator and provided input on how the network simulator features worked, and guidance on writing a paper.

Relation to Research Questions

This paper addresses RQ1 and RQ2.

4.4 Relation between Papers

Figure 4.1 shows how the papers relate to the R&D phases. Paper A analyses the different R&D phases. Paper B relates to software architecture through the Requirement, Design and Implementation phase. Paper C relates to improved efficiency when moving between Research to Verification phase.

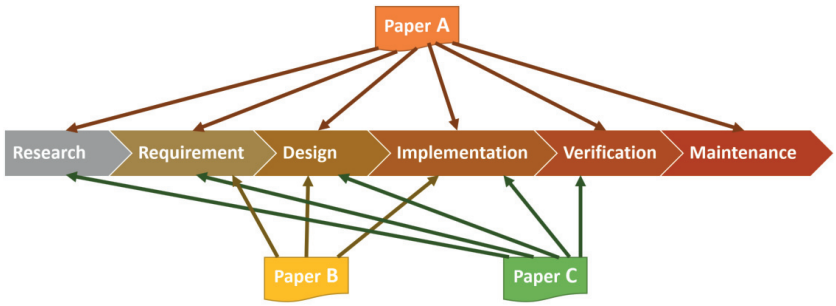

Figure 4.1: Papers in relation to R&D phases

Figure 4.2 shows how the papers relate to the challenges. Paper A identifies the challenges related to industrial communication software. Paper B proposes a communication stack design that addresses the challenges DCS, Changeability and Reuse. Paper C describes a custom simulation based on the stack design in Paper B, and in addition addresses the challenges, Granularity and Debugging.

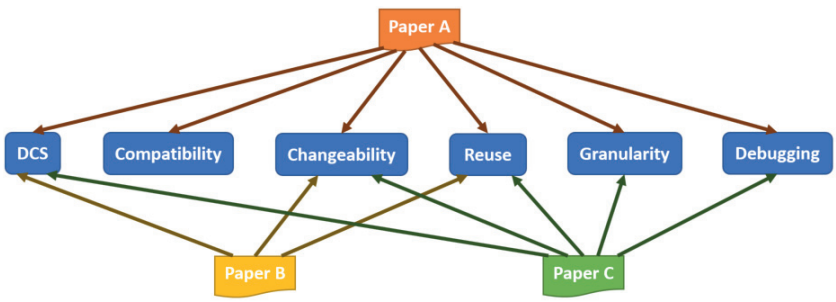

Figure 4.2: Papers in relation to identified challenges.

Chapter 5

Conclusions

Development of distributed systems is hard. Development of time critical industrial distributed control systems is even harder. Although many challenges related to the development of communication software are common for most networked systems, there are still challenges to resolve for industrial DCS, mainly because they are time critical. This thesis has highlighted the need to increase the efficiency and reuse of information between the intermediate phases when going from research to deployment, as well as proposed solutions to facilitate efficient transitions between the R&D phases. Therefore, a new approach is proposed in this thesis to automate, simplify and reduce unnecessary work throughout the R&D phases. The introduction of a new type of simulator which is simple to implement, maintain, extend, and use, bridge the gaps in and between R&D phases and extending the use of simulation in the overall R&D workflow. The findings from the proof of concept implementation applied as a case study in a research project, indicate that the identified simulator functionalities are improving the R&D efficiency. Especially the simplicity to change between simulations and the real testbed using the same code was found beneficial by the R&D team, as the transition was perceived more or less as seamless and indirectly contributed to the overall efficiency. As with other network evaluation methods and tools, the proposed approach with a customized simulator will not resolve or be applicable for all needs. It does however show that it is feasible to extend the use of simulation when developing industrial DCS, and facilitate the efficiency when moving from the research to the verification phase, with rapid iterations to earlier phases.

5.1 Research Questions Revisited

Regarding RQ1 and RQ2, the proof of concept in Paper C made the transitions between the simulator and the testbed easy, since the simulator and target system share the code base as well as the integrated development environment. In addition, the same development tools were available in both the testbed and the simulator which further enforced the usability and ease-of-use. The main difference in the workflow was that the simulator allows total control over certain important network behaviors, while proper timing was ensured on the target system. The identified subset of simulator functionality, especially time synchronization between RTOS and communication media, symmetric and asymmetric links, packet loss, and synchronized break points, proved to be useful for evaluating research theories, debugging, finding logical errors and wrong assumptions in the research project. The drawback with making a customized simulator is that it is limited to a specific platform, and that modifications to other platform parts may be necessary. However the approach to adapt the simulator to the reality at hand indicate that a customized simulator that covers more than one R&D phase may be a good alternative to more advanced simulators to increase the overall R&D efficiency. Another important finding was the MAC filter functionality that was introduced into the target system code, in order to create specific mesh topologies in the testbed that was almost impossible to create even by careful manual placement of the nodes in the premises. It did however not provide full control of the topology, nor did it provide control over disturbances, but was useful for evaluating the correct behavior and timing of the real-time protocol stack given specific topologies. Easy switching between the simulator and the testbed improved the overall efficiency of the R&D project, since wrong assumptions and logical errors could be found and eliminated without using the testbed. In addition the new approach enabled an efficient problem solving strategy with reduced efforts. For example if a problem only exists in the testbed, it was most likely related to hardware resources or system timing.

Regarding RQ3, the proof of concept implementation described in Paper B, shows that it is possible to support different types of communication protocols and run-time contexts with the same communication stack design and implementation. Both open source and COTS communication stacks for resource constrained embedded systems commonly support execution with or without an OS. However, to my knowledge, no communication stack design can support either real-time or non-real-time communication protocols only by applying different resource allocations strategies. Although the design enables

improved code reuse and changeability, it is unlikely to have a huge impact, since it is more efficient to use a COTS or an open source stack for already existing protocol implementations. This design did however make the implementation of the customized simulator simple, especially since the lower simulation layers in different stacks could be driven by the same thread. The contributions in this thesis are addressing a subset of the challenges identified in Paper A, e.g., changeability, reuse, granularity and debugging, when developing and maintaining industrial DCS. However, there are still a lot of work remaining, related to the research questions and the overall goal to streamline the workflow from research to deployment of industrial DCS.

5.2 Future Work

The work so far has focused on the combination of network simulators and testbeds. Hence a more detailed study of the alternatives such as virtualization, emulation and instruction level simulation is needed to get a more comprehensive view of all the alternatives and their usefulness to R&D on industrial DCS. In addition, the contributions have been on communication software for time sensitive homogeneous networks, hence there is a need to explore the effects on heterogeneous networks with different characteristics and needs. As systems become more complex, there is a need for further automation of manual and error prone tasks, and simplification of the use of methods and tools throughout the research and development cycles.

Bibliography

[1] ABB. System 800xA Solutions Handbook. *Doc No: 3BSE069330 Revision: C Language: English*, 2013.

[2] Barry Boehm and V.R. Basili. Software Defect Reduction Top 10 List. *Computer*, 34(1):135–137, 2001.

[3] D. Hallmans, M. Jagemar, S. Larsson, and T. Nolte. Identifying evolution problems for large long term industrial evolution systems. *Proceedings - IEEE 38th Annual International Computers, Software and Applications Conference Workshops, COMPSACW 2014*, pages 384–389, 2014.

[4] Dr. Winston W. Royce. Managing the Development of large Software Systems. *Ieee Wescon*, (August):1–9, 1970.

[5] Mohammad Sharif and Abolghasem Sadeghi-Niaraki. Ubiquitous sensor network simulation and emulation environments: A survey. *Journal of Network and Computer Applications*, 93(May):150–181, 2017.

[6] Alexander Zimmermann, Mesut Gunes, Martin Wenig, Ulrich Meis, and Jan Ritzerfeld. How to Study Wireless Mesh Networks: A hybrid Testbed Approach. In *21st International Conference on Advanced Information Networking and Applications*, pages 853–860, Niagara Falls, ON, 2007. IEEE.

[7] Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt. Cross-level sensor network simulation with COOJA. *Proceedings - Conference on Local Computer Networks, LCN*, pages 641–648, 2006.

[8] Joakim Eriksson. *Detailed Simulation of Heterogeneous Wireless Sensor Networks*. PhD thesis, Uppsala University, 2009.

[9] Lei Shu, Manfred Hauswirth, Han-Chieh Chao, Min Chen, and Yan Zhang. NetTopo: A framework of simulation and visualization for wireless sensor networks. *Ad Hoc Networks*, 9(5):799–820, 2011.

[10] Sung Park, Andreas Savvides, and Mani B Srivastava. SensorSim: A Simulation Framework for Sensor Networks. In *Proceedings of the 3rd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems - MSWIM '00*, pages 104–111, New York, New York, USA, 2000. ACM Press.

[11] Kristian Sandström, Aneta Vulgarakis, Markus Lindgren, and Thomas Nolte. Virtualization technologies in embedded real-time systems. *IEEE International Conference on Emerging Technologies and Factory Automation, ETFA*, 2013.

[12] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, Alex Ho, Rolf Neugebauer, Ian Pratt, and Andrew Warfield. Xen and the Art of Virtualization. *ACM SIGOPS Operating Systems Review*, 37(5):164–177, 2003.

[13] S Balaji. Waterfall vs v-model vs agile : A comparative study on SDLC. *WATEERFALL Vs V-MODEL Vs AGILE : A COMPARATIVE STUDY ON SDLC*, 2(1):26–30, 2012.

[14] Victor R Basili. Reusing Existing Software. Technical report, Institute for Advanced Computer Studies, University of Maryland, Collage Park, Maryland, 1988.

[15] Barry W. Boehm, T R W Defense, Systems Group, Harry W. Boehm, T R W Defense, and Systems Group. A Spiral Model of Software Development and Enhancement. *Computer*, 21(May):61–72, 1987.

[16] Barry W Boehm, Chris Abts, and Sunita Chulani. Software development cost estimation approaches — A survey. *Annals of Software Engineering*, 10(1-4):177–205, 2000.

[17] Mary Shaw. What makes good research in software engineering? *International Journal on Software Tools for Technology . . .* , 4(1):1–7, 2002.

[18] M Reiser. A Queueing Network Analysis of Computer Communication Networks with Window Flow Control. *IEEE Trans. on Communications*, 27(8):1199–1209, 1979.

[19] Tronje Krop, Michael Bredel, Matthias Hollick, and Ralf Steinmetz. JiST/MobNet: combined simulation, emulation, and real-world testbed for ad hoc networks. In *Proceedings of the Second ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation and Characterization*, pages 27–34, New York, New York, USA, 2007. ACM Press.

[20] K.M. Chandy and Jayadev Misra. Distributed Simulation: A Case Study in Design and Verification of Distributed Programs. *IEEE Transactions on Software Engineering*, SE-5(5):440–452, 1979.

[21] Richard M. Fujimoto. Parallel discrete event simulation. *Communications of the ACM*, 33(10):30–53, oct 1990.

[22] Xiang Zeng, Rajive Bagrodia, and Mario Gerla. GloMoSim: A Library for Parallel Simulation of Large-scale Wireless Networks Xiang. In *Proceedings. Twelfth Workshop on Parallel and Distributed Simulation PADS '98 (Cat. No.98TB100233)*, pages 154–161, Banff, Alta, 1998. IEEE Comput. Soc.

[23] Muhammad Imran, Abas Md Said, and Halabi Hasbullah. A survey of simulators, emulators and testbeds for wireless sensor networks. *Proceedings 2010 International Symposium on Information Technology - Engineering Technology, ITSim'10*, 2:897–902, 2010.

[24] Kevin Fall. Network emulation in the VINT/NS simulator. In *Proceedings IEEE International Symposium on Computers and Communications (Cat. No.PR00250)*, pages 244–250. IEEE Comput. Soc, 1999.

[25] Marta Carbone and Luigi Rizzo. Dummynet Revisited. *ACM SIGCOMM Computer Communication Review*, 40(2):12, 2010.

[26] Thomas Staub, Reto Gantenbein, and Torsten Braun. VirtualMesh: an emulation framework for wireless mesh networks in OMNeT++. In *Proceedings of the Second International ICST Conference on Simulation Tools and Techniques*. ICST, 2009.

[27] Klaus Wehrle, Elias Weingärtner, and Hendrik vom Lehn. Device Driver-enabled Wireless Network Emulation. In *Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques*, pages 188–197. ACM, 2011.

[28] Elias Weing, Florian Schmidt, Hendrik Lehn, Tobias Heer, and Klaus Wehrle. SliceTime : A platform for scalable and accurate network emulation. In *8th USENIX Symposium on Networked Systems Design and Implementation*, Boston, MA, USA, 2011.

[29] N.M.M.K. Chowdhury and R Boutaba. Network virtualization: state of the art and research challenges. *Communications Magazine, IEEE*, 47(7):20–26, 2009.

[30] Bob Lantz and Brian O'Connor. A Mininet-based Virtual Testbed for Distributed SDN Development. *ACM SIGCOMM Computer Communication Review*, 45(5):365–366, 2015.

[31] Jeff Ahrenholz, Claudiu Danilov, Thomas R Henderson, and Jae H Kim. CORE: A real-time network emulator. In *Proceedings - IEEE Military Communications Conference MILCOM*, pages 1–7, 2008.

[32] Chengchao Liang and F. Richard Yu. Wireless Network Virtualization: A Survey, Some Research Issues and Challenges. *IEEE Communications Surveys & Tutorials*, 17(1):358–380, 2015.

[33] Imran Khan, Fatna Belqasmi, Roch Glitho, Noel Crespi, Monique Morrow, and Paul Polakos. Wireless Sensor Network Virtualization: A Survey. *IEEE Communications Surveys & Tutorials*, 18(1):553–576, 2016.

[34] Emmanuel Baccelli, Oliver Hahm, Mesut Gunes, Matthias Wahlisch, and Thomas Schmidt. RIOT OS: Towards an OS for the Internet of Things. *2013 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 79–80, 2013.

[35] Ludwig Ortmann. *Virtualization of the RIOT Operating System*. PhD thesis, Freie Universität, 2015.

[36] Brent Chun, David Culler, and Timothy Roscoe. PlanetLab: An Overlay Testbed for Broad-coverage Services. *ACM SIGCOMM Computer Communication Review*, 33(3):3–12, 2003.

[37] R N Murty, G Mainland, I Rose, A R Chowdhury, A Gosain, J Bers, and M Welsh. CitySense: An Urban-Scale Wireless Sensor Network and Testbed. *Technologies for Homeland Security, 2008 IEEE Conference on*, pages 583–588, 2008.

[38] Olof Rensfelt, Frederik Hermans, Christofer Ferm, Per Gunningberg, and L. Larzon. Sensei-UU: A Nomadic Sensor Network Testbed Supporting Mobile Nodes. 2009.

[39] Anne-Sophie Tonneau, Nathalie Mitton, and Julien Vandaele. A Survey on (mobile) Wireless Sensor Network Experimentation Testbeds. In *2014 IEEE International Conference on Distributed Computing in Sensor Systems*, pages 263–268. IEEE, may 2014.

[40] Geoffrey Werner-Allen, Patrick Swieskowski, and Matt Welsh. MoteLab: A Wireless Sensor Network Testbed. In *IPSN 2005. Fourth International Symposium on Information Processing in Sensor Networks, 2005.*, pages 483–488. IEEE, 2005.

[41] David Johnson, Tim Stack, Russ Fish, Dan Flickinger, Robert Ricci, and Jay Llepreau. TrueMobile: A Mobile Robotic Wireless and Sensor Network Testbed. *University of Utah Flux Group Technical Note FTN–2005–02*, 2005.

[42] Andras Varga. The OMNeT++ Discrete Event Simulation System. In *Proceedings of the European Simulation Multiconference*. IEEE, dec 2001.

[43] Thomas R Henderson and George F Riley. Network Simulations with the ns-3 Simulator. *Sigcomm'08*, page 527, 2006.

[44] P.S. Magnusson, Magnus Christensson, Jesper Eskilson, Daniel Forsgren, G. Hallberg, J. Hogberg, Fredrik Larsson, Andreas Moestedt, and Bengt Werner. Simics: A full system simulation platform. *Computer*, 35(2):50–58, 2002.

[45] J Engblom, D Kagedal, A Moestedt, and J Runeson. Developing Embedded Networked Products using the Simics Full-System Simulator. In *2005 IEEE 16th International Symposium on Personal, Indoor and Mobile Radio Communications*, volume 2, pages 785–789. IEEE, 2005.

[46] Jonathan Polley, Dionysys Blazakis, J. McGee, Dan Rusk, J.S. Baras, and Manish Karir. ATEMU: a fine-grained sensor network simulator. In *2004 First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks, 2004. IEEE SECON 2004.*, pages 145–152. IEEE, 2004.

[47] Fabrice Bellard. QEMU , a Fast and Portable Dynamic Translator. *USENIX Annual Technical Conference. Proceedings of the 2005 Conference on*, pages 41–46, 2005.

[48] Ben L. Titzer, Daniel K. Lee, and Jens Palsberg. Avrora: Scalable Sensor Network Simulation with Precise Timing. In *Proceedings of the 4th International Symposium on Information Processing in Sensor Networks*, 2005.

[49] Philip Levis, Nelson Lee, Matt Welsh, and David Culler. TOSSIM: Accurate and Scalable Simulation of Entire TinyOS Applications. In *Proceedings of the first international conference on Embedded networked sensor systems - SenSys '03*, pages 126–137, New York, New York, USA, 2003. ACM.

[50] Johan Åkerberg, Mikael Gidlund, and Mats Björkman. Future Research Challenges in Wireless Sensor and Actuator Networks Targeting Industrial Automation. In *2011 9th IEEE International Conference on Industrial Informatics*, pages 410–415, Caparica, Lisbon, Portugal, 2011. IEEE.

[51] Daniel Camara, Hajime Tazaki, Emilio Mancini, Thierry Turletti, Walid Dabbous, and Mathieu Lacage. DCE: Test the real code of your protocols and applications over simulated networks. *IEEE Communications Magazine*, 52(3):104–110, 2014.

[52] Sam Jansen and Anthony McGregor. Simulation with real world network stacks. In *Proceedings - Winter Simulation Conference*, volume 2005, pages 2454–2463, 2005.

[53] Adam Dunkels. Design and Implementation of the lwIP TCP/IP Stack. *Swedish Institute of Computer Science*, 2001.

[54] Douglas C. Schmidt. The ADAPTIVE Communication Environment. 32:214—-225, 1993.

[55] Douglas C Schmidt and Charles D Cranor. An Architectural Pattern for Efficient and Well-structured Concurrent I / O. *Structure*, pages 1–11, 1996.

[56] D Schmidt. Acceptor and Connector - Design Patterns for Initializing Communication Services. *EuroPloP*, pages 1–13, 1996.

[57] Douglas C Schmidt. Reactor: An Object Behavioral Pattern for Demultiplexing and Dispatching Handles for Synchronous Events 2 Also Known As 3 Example. *Pattern Languages of Programs*, pages 1–11, 1994.

[58] R Greg Lavender and Douglas C Schmidt. Active object: an object behavioral pattern for concurrent programming. *Pattern languages of program design 2*, pages 483–499, 1996.

[59] Jason Hill, Robert Szewczyk, Alec Woo, Seth Hollar, David Culler, and Kristofer Pister. System Architecture Directions for Networked Sensors. *ACM SIGARCH Computer Architecture News*, 28(5):93–104, dec 2000.

[60] Adam Dunkels, Björn Grönvall, and Thiemo Voigt. Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors. In *29th Annual IEEE International Conference on Local Computer Networks*, pages 455–462. IEEE, 2004.

[61] David Gay, Philip Levis, and David Culler. Software design patterns for TinyOS. *ACM Transactions on Embedded Computing Systems*, 6(4):40–49, 2007.

[62] Zhibo Pang, Kan Yu, Johan Åkerberg, and Mikael Gidlund. An RTOS-based architecture for industrial wireless sensor network stacks with multi-processor support. In *Proceedings of the IEEE International Conference on Industrial Technology*, pages 1216–1221, 2013.