Scalable and Interoperable Low-Power Internet of Things Networks

JOAKIM ERIKSSON
Dissertation presented at Uppsala University to be publicly examined in Häggsalen, Ångströmlaboratoriet, Lägerhyddsvägen 1, Uppsala, Friday, 15 December 2023 at 13:15 for the degree of Doctor of Philosophy. The examination will be conducted in English. Faculty examiner: Professor Leo Selavo (University of Latvia).

Abstract


Internet of Things (IoT) is the concept of connecting devices to the Internet. IoT devices can be anything from small temperature sensors to self-driving cars. The devices are typically resource-constrained, connected wirelessly, and often battery-powered. In this thesis, we address energy efficiency and the tools required for estimating power consumption, interoperability between different implementations of IoT protocols, and scalability of the IoT networks in mesh configurations. The contributions are made in the five included research papers addressing these challenges. Firstly, we present and evaluate network-wide energy estimation support in our simulation tool COOJA/MSPSim. Due to the timing accuracy of the simulation and emulation, we get energy consumption estimates very close to hardware-based estimates. The second contribution evaluates the capabilities of simulation tools for interoperability testing. We show that it is possible to set up simulations of networks with multiple implementations of the same open standards (6LoWPAN/RPL) and that it is possible to get results beyond pure interoperability, including power consumption and network quality. Finally, we show that, by carefully managing neighbor updates, it is possible to scale IoT networks even when the IoT devices' memory limitations severely constrain the size of the neighbor table.

The experimental systems research that resulted in this thesis also provided significant contributions to the open-source ecosystem around Contiki, an operating system for resource-constrained IoT devices. This software, Contiki and COOJA/MSPSim, has been a cornerstone in our capability to perform sound systems research and has been widely used by other research groups in resource-constrained IoT research in academia and many companies for developing commercial IoT devices.

Keywords: IoT, low-power networking, scalability, interoperability

Joakim Eriksson, Department of Electrical Engineering, Networked Embedded Systems, Box 65, Uppsala University, SE-751 03 Uppsala, Sweden.

© Joakim Eriksson 2023

ISSN 1651-6214
URN urn:nbn:se:uu:diva-513926 (http://urn.kb.se/resolve?urn=urn:nbn:se:uu:diva-513926)
To Viktor, Oliver and Agneta
List of papers

This thesis is based on the following papers, which are referred to in the text by their Roman numerals.


Reprints were made with permission from the publishers.
Additional Publications

List of selected additional publications not included in the Thesis:


- John Kanwar, Niclas Finne, Nicolas Tsiftes, Joakim Eriksson, Thiemo Voigt, Zhitao He, Christer Åhlund, and Saguna Saguna "JamSense: Interference and Jamming Classification for Low-power Wireless Networks". In *13th IFIP Wireless and Mobile Networking Conference (WMNC)*, 2021

- Eric Samikwa, Thiemo Voigt, and Joakim Eriksson, "Flood prediction using IoT and artificial neural networks with edge computing". In *International Conferences on Internet of Things (iThings)*, 2020


- Simon Duquennoy, Joakim Eriksson and Thiemo Voigt, "Five-Nines Reliable Downward Routing in RPL". In *arXiv 2017*

- Oriol Piñol Piñol, Shahid Raza, Joakim Eriksson, and Thiemo Voigt, "BSD-based elliptic curve cryptography for the open Internet of Things"
• Adam Dunkels, Joakim Eriksson, Niclas Finne, Fredrik Österlind, Nicolas Tsiftes, Julien Abeillé, and Mathilde Durvy "Low-power IPv6 for the Internet of Things". In Ninth International Conference on Networked Sensing (INSS), 2012

• Jeonggil Ko, Joakim Eriksson, Nicolas Tsiftes, Stephen Dawson-Haggerty, Andreas Terzis, Adam Dunkels, and David Culler "Contikirpl and tinyrpl: Happy together". In Workshop on Extending the Internet to Low Power and Lossy Networks, Workshop at IPSN 2011


• Nicolas Tsiftes, Joakim Eriksson, and Adam Dunkels, "Low-power wireless IPv6 routing with ContikiRPL." In International Conference on Information Processing in Sensor Networks (IPSN), 2010

• Niclas Finne, Joakim Eriksson, Nicolas Tsiftes, Adam Dunkels, and Thiemo Voigt, "Improving Sensornet Performance by Separating System Configuration from System Logic". In European Conference on Wireless Sensor Networks (EWSN), 2010


• Adam Dunkels, Niclas Finne, Joakim Eriksson, and Thiemo Voigt, "Runtime dynamic linking for reprogramming wireless sensor networks". In International Conference on Embedded Networked Sensor Systems ( Sen-Sys), 2006

• Fredrik Österlind, Adam Dunkels, Joakim Eriksson, Niclas Finne, and Thiemo Voigt, "Cross-level sensor network simulation with COOJA". In IEEE Conference on Local Computer Networks (LCN), 2006
Contents

Part I: Dissertation Summary ............................................................................................... 13

1 Introduction ......................................................................................................................... 15
  1.1 Low-Power Wireless Networking for Internet of Things .............................................. 15
    1.1.1 Application Areas ................................................................................................. 16
    1.1.2 Communication Protocols for IoT Devices ......................................................... 16
    1.1.3 The Internet Protocols for IoT ........................................................................... 16
  1.2 Resource-Constrained Devices .................................................................................... 17
    1.2.1 Embedded Operating Systems ......................................................................... 17
    1.2.2 Emulation and Simulation ................................................................................. 18
  1.3 Research Methodology ................................................................................................. 20
  1.4 Dissertation Structure .................................................................................................. 20

2 Research Challenges .......................................................................................................... 21
  2.1 Power Profiling and Energy Efficiency ....................................................................... 21
  2.2 IoT Network Interoperability ..................................................................................... 21
  2.3 IoT Network Scalability .............................................................................................. 22

3 Contributions ................................................................................................................... 23
  3.1 Operating System and Simulation Support for Network-Scale Power Profiling ............ 23
  3.2 Interoperability and Performance Testing for IoT Stacks .......................................... 24
  3.3 Network Scalability under Memory Constraints ....................................................... 24
  3.4 Software Contributions .............................................................................................. 25

4 Summary of the Papers ..................................................................................................... 26
  4.1 Paper I: Accurate, Network-Scale Power Profiling for Sensor Network Simulators ...... 26
  4.3 Paper III: Industry: Beyond Interoperability - Pushing the Performance of Sensor Network IP Stacks ................................................................. 28
  4.4 Paper IV: The Contiki-NG Open Source Operating System for Next Generation IoT Devices ........................................................................................................ 29
  4.5 Paper V: Scaling RPL to Dense and Large Networks with Constrained Memory ........ 30

5 Related Work .................................................................................................................... 32
  5.1 Operating Systems ....................................................................................................... 32
5.2 Simulation Tools ................................................................. 33
5.3 Energy Estimation ............................................................. 35
5.4 Automated Interoperability Testing ..................................... 36
5.5 Scalability of IP-based IoT Mesh Networks ....................... 37

6 Conclusions and Future Work ................................................. 38
  6.1 Conclusions ................................................................. 38
  6.2 Future Work ............................................................... 39

7 Summary in Swedish .......................................................... 41

References ............................................................................. 43
Personal Acknowledgements

Finally, after more than 20 years of being a Ph.D. student, I am at the stage of writing the Acknowledgements. When I started at the Swedish Institute of Computer Science (SICS) as a master’s thesis student, I decided to jump out of the University and join a research institute. Little did I know that I would be working there more than 25 years later and writing a Ph.D. thesis Acknowledgments section, but I am happy that I do. There are so many people that I have been working with during this time; I will include the ones I believe are the most essential part of my success in getting a Ph.D.

First and foremost, my deepest gratitude to my primary supervisor, Professor Thiemo Voigt, who has been guiding me forward with infinite patience. I would also like to show my gratitude to my current co-supervisor Professor Per Gunningberg. He was one of my teachers at Uppsala University during my computer science studies and also helped me find my way into SICS via a master’s thesis project. During the years as a Ph.D. student, I have had several other supervisors and co-supervisors, I am grateful for all their support over the years, Professor Mats Björkman, Professor Arne Andersson, Professor Erik Aurell, Professor Mikael Gidlund.

Another important person in my life as a Ph.D. student and researcher at SICS and now RISE is Dr. Sverker Janson, who offered an exciting master’s thesis project and, after that, allowed me to work in his research lab, and was my manager for over two decades. I would also like to thank my current manager, Dr. Hanifeh Khayyeri, for supporting me while working on finalizing my thesis.

Some years into my employment at SICS, we created the Network Embedded Systems group with a strong focus on sensor networks and IoT. Our research group has been very successful and is also a strong reason for my success in getting a Ph.D. I would like to thank all the current and previous researchers in our group, including Dr. Nicolas Tsiftes, Niclas Finne, Dr. Adam Dunkels, Professor Luca Mottola, Dr. Simon Duquennoy, Dr. Fredrik Rosendal, and Joel Höglund.

Other colleagues at RISE I would like to thank for exciting discussions and projects include Professor Seif Haridi, Dr. Mats Carlsson, Dr. Per Mildner, Dr. Daniel Gillblad, Dr. Rebecca Steinert, Dr. Ahmad Al-Shishtawy, Professor Jim Dowling, Professor Dejan Kostic, Professor Kristina Höök, Dr. Anna Ståhl, and Dr. Anton Gustafsson.

Last but not least, I would like to thank my family, Agneta, Oliver, and Viktor, for their understanding and support while writing the thesis, and my parents, Sylvia and Ruben, who supported me throughout my journey of becoming and being a computer science student.
Funding Acknowledgements
While conducting the research for this thesis, I have been involved in a large number of research projects, most of which were supported through public funding. I am very grateful to all the funding agencies for financing the research. The agencies include Swedish Agency for Innovation Systems (VINNOVA), Swedish Foundation for Strategic Research (SSF), and the Swedish Knowledge Foundation (KKS), and European Commission under the FP7. Finally I am also grateful to Yanzi Networks, a Swedish SME, that funded most of the work resulting in Paper V, where we worked on commercializing Contiki-based IoT products and services. When working on this thesis, I was employed as a researcher by RISE, Research Institutes of Sweden, and before that by SICS, Swedish Institute of Computer Science AB (merged into RISE).

Joakim Eriksson
Solna, October 2023
Part I:  
Dissertation Summary
1. Introduction

The Internet of Things (IoT) is the concept of connecting devices to the Internet. IoT devices come in a wide variety and include smartphones, home appliances, industrial sensors, actuators, and vehicles. A complete IoT system also includes everything from the IoT devices to cloud services, including the gateway that connects devices to the Internet. In many cases, the IoT devices form a mesh network and communication terminates at a gateway that either hosts an application with local logic or forwards the data to the cloud, often after protocol translation. IP-based IoT connectivity provides end-to-end connection, without protocol translation, to the cloud, thus providing a possibility of true end-to-end security.

This thesis focuses on system software for low-power IoT devices and IP-based mesh networks between the devices and the gateway.

1.1 Low-Power Wireless Networking for Internet of Things

Low-power wireless networking is used for a large number of different types of applications such as remote sensing and monitoring, smart office sensing and actuation, and wearables. These networks can have different topologies, typically either a star topology where every device connects via a single hop to a central gateway, or a mesh topology where many nodes can forward packets and create a longer-range network via multiple hops to the gateway. Different applications use different communication protocols and topologies. Star networks such as BLE [9] are often used for single hop, short-range networks such as smart home appliances and wearables, while LoRa/LoRaWAN [4] and mioty [22] are used in long-range single-hop networks such as farming, smart cities, and monitoring of remote equipment. More complex mesh networks are used in some smart building and smart home applications (Thread [25] or Zigbee [72]) using short-range links with IEEE 802.15.4 on 2.4 GHz, while smart grids and smart cities use longer-range links on sub-GHz (Wi-Sun [26]).

Typical devices for low-power wireless networking consist of a system-on-chip with a microprocessor (MCU), built-in radio, and digital and analog I/Os. These devices are usually powered by batteries and are often designed to have an expected battery lifetime of up to ten years [53].

To achieve ten years of battery lifetime devices are, as much as possible, in deep-sleep modes that consume a very small amount of power. They only
wake up regularly to measure, process and communicate during a short period (typically below 100 ms). In the simplest form of network topology where everything is single-hop, all the devices can have very long sleep times as there is no need to route packets to other nodes. The situation in mesh networks is much more complex as many nodes need to route for other nodes further from the gateway. In a mesh network, the routing nodes either need to be mains powered or make use of a duty-cycling protocol such as IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH) [30] that allow the radio and the microprocessor to be in sleep states for most of the time but still wake up at short intervals to forward packets on behalf of neighboring nodes.

1.1.1 Application Areas

IoT is used in many types of application areas where remote sensing and actuation are needed. Infrastructure monitoring is one case where sensors are deployed on bridges, buildings, or the electrical grid to monitor the condition of the infrastructure to enable predictive maintenance [49]. Another area where IoT is used is in smart buildings [3, 36] where sensors provide detailed information about indoor air quality (temperature, $CO_2$ and particles), and energy consumption. These IoT systems provide building users with data for better indoor environment control and optimization of the building’s energy consumption. IoT is also used in many other areas, including wearables for personal health and medical applications, sensors and actuators for smart homes, and condition monitoring of industrial equipment.

1.1.2 Communication Protocols for IoT Devices

IoT devices are connected to the gateway and/or cloud via several different protocols. High-performance IoT devices, such as network cameras and wireless audio speakers can be connected via WiFi or even Ethernet. Low power and more resource-constrained devices such as battery-powered sensors typically use low-power communication protocols such as 6LoWPAN [52], BLE, Z-Wave, Zigbee or LoRa. In the near future, it seems like the low-power version of telecom protocols will also be viable options, especially NB-IoT (Narrowband IoT) that is designed to have a lower bitrate and significantly lower power consumption than the previous telecom alternatives [1, 2].

1.1.3 The Internet Protocols for IoT

In 2005, the first draft of 6LoWPAN communication was submitted to the IETF, the Internet Engineering Task Force, the organisation that handles standardisation of the Internet Protocols. Now, there are many completed RFCs from the initial 6LoWPAN IPHC, IPv6 header compression for IEEE802.15.4
to the more recent 6TiSCH for reliable low-power wireless connectivity [51, 67, 69]. Recently the 6LoWPAN-based Thread [68] protocol established itself as one of the likely candidates for being the main alternative for low-power IoT devices in smart homes.

1.2 Resource-Constrained Devices
The work in this thesis revolves around highly resource-constrained devices. A typical device is the Telos B [57], also called Tmote Sky, with a 16-bit MSP430 MCU with 10 kB RAM and 48 kB Flash, an 802.15.4 radio, and a few sensors. The Tmote Sky was, for a long time, the most common platform for research within wireless sensor networks. Recently, more and more devices contain a system-on-chip where MCU and radio-chip are integrated into one single chip, typically with more memory than the Tmote Sky and a 32-bit ARM cortex M3 MCU. A study by Ko et al. that analyses the shift from 8/16-bit to 32-bit shows that the expected increase in power consumption and usage of RAM/flash when upgrading to 32-bits is not that big. This result indicates that it is likely that most future low-power products will use 32-bit MCUs [40]. Even with the move to 32-bit MCUs, the flash and RAM will be limited, and given these constraints, devices cannot use a fully featured operating system such as Linux but need a more lightweight operating system (OS). The work in this thesis has been performed on this type of resource-constrained devices using Contiki OS as the operating system.

1.2.1 Embedded Operating Systems
Even for very resource-constrained IoT devices it is important to develop applications on top of an OS. Like Linux or Windows, the embedded OSes support memory management, processes, networking, and peripherals such as GPIO, sensors, and radios. This section introduces the OSes that are most relevant to the thesis.

Contiki [15] is a very lightweight event-based operating system with support for many IoT device platforms with built-in 802.15.4 radio. Contiki also supports a full IPv6 stack with 6LoWPAN compression [17], multi-hop mesh networking, and many application layer protocols including HTTP, MQTT, and LWM2M. During the work resulting in this thesis, I contributed to Contiki’s 6LoWPAN stack and maintained the Tmote Sky platform. Contiki and the full IPv6 stack, including an application, fit the 50 kB flash of the Tmote Sky, making it one of the smallest operating systems with a full IPv6 stack. In 2017, we released Contiki-NG (Contiki-Next Generation, described in Paper IV), a fork with a focus on standards-based communication and support for modern IoT devices.
TinyOS [28] was one of the first OSes for sensor networks and IoT. It is an event-driven OS designed for resource-constrained hardware such as the Tmote Sky and made use of a component-oriented C-language extension. TinyOS’s IPv6 stack, BLIP, implements 6LoWPAN compression and supports RPL-routing via TinyRPL [39]. We used both TinyOS and Contiki to evaluate automatic interoperability testing as described in Paper II and Paper III.

1.2.2 Emulation and Simulation

A simulator is a software tool that simulates selected parts of the behavior of the real world and is normally used as a tool for research and development. Simulators exist in a wide variety of fields, including physics, biology, economics, and computer systems. Depending on the intended usage of the simulator, different parts of the real-world system are modeled. The parts that are modeled can also be of varying abstraction levels. A wireless sensor network simulator simulates the wireless media and the nodes in the network. Some sensor network simulators have a detailed model of the wireless media, including the effects of obstacles between nodes [37], while others have a more abstract model [45]. In many wireless sensor simulators, the focus is on the node models where the nodes can be simulated in detail, in some cases using hardware emulation for realistic execution of the node’s firmware. In this thesis, the focus is mainly on simulators that have detailed node models.

During the development of applications, systems, and protocols for sensor networks, a large part of the time is spent compiling, testing, debugging and evaluating. Either a network of real sensor nodes or a wireless sensor network simulator is used during the testing, debugging and evaluation. Besides Contiki, our research group also develops simulation tools such as COOJA/MSPSim [19, 55]. By using COOJA/MSPSim we have found both alignment problems and compiler bugs during the development and porting of communication stacks and other software. COOJA/MSPSim is also used for Contiki-NGs automatic tests at GitHub to enable full system testing.

Research experiments for sensor networks and IoT are either deployed and evaluated in a testbed with real devices or in a simulator [24, 31]. When using simulators, the evaluation can be much faster, and information about nodes and their communication can be measured at a high level of detail. It is also possible to repeat the exact same experiment several times, something more or less impossible when evaluating an experiment in a testbed with real devices. In a simulation, it is also possible to control most aspects of the environment, including the number of nodes, mobility, and packet loss ratio. The main problem with simulators is that they abstract away aspects of the real world, which means that they will never fully replace testing in testbeds to validate the simulation results. An example of a limitation in our simulator is COOJA’s quite simple radio propagation models.
The term emulator is another name for an instruction set simulator. An emulator allows compiled code, the firmware, to be executed without modifications and therefore not only simulates the effect of using a protocol, but simulates the effect of using the specific protocol implementation, compiled with the same compiler, and using the same radio-driver that will be used on the actual device when deployed.

Some of the work in this thesis relies on the COOJA/MSPSim Simulator. COOJA/MSPSim combines two tools into a wireless sensor network simulator that allows execution of fast nodes compiled for the host machine (e.g., x86/linux) and of slower emulated nodes running the same firmware as the real IoT device would run (compiled msp430 software). This ability of COOJA is called cross-level simulation [55]. The following sections introduce these tools in more detail.

MSPSim [19] is a Java-based instruction-level emulator of the MSP430 microprocessor series. It emulates complete sensor networking platforms such as the Tmote Sky [57] and ESB/2 [62]. MSPSim provides detailed simulation with accurate timing and strong debugging support. MSPSim combines cycle-accurate interpretation of MCU instructions with a discrete-event-based simulation of all other components, both internal and external. MSPSim uses an event-based execution kernel that enables accurate timing while limiting the host processor’s utilization. MSPSim provides debugging capabilities such as breakpoints, watches, logging, and single-stepping as well as statistics about the operating modes of the emulated components. These statistics include elapsed processing time in the different low-power modes that can be used for energy consumption estimations. Users can access all features and information via a command line interface or Java APIs. I initiated the MSPSim project and have been the leading developer of the emulation platform.

COOJA is a flexible Java-based network simulator initially designed for simulating networks of IoT nodes running the Contiki operating system. In a COOJA simulation, each node can be of a different type; differing not only in onboard software but also in the simulated hardware. COOJA is flexible in that many parts of the simulator can be easily replaced or extended with additional functionality. COOJA can execute Contiki programs by running the program code compiled for the desktop host CPU or running code compiled for the IoT device in MSPSim. COOJA can also run nodes programmed in Java. The different approaches have advantages and disadvantages. Java-based nodes enable much faster simulations but do not run deployable code. Emulating nodes allows control and retrieval of more fine-grained execution details compared to Java-based nodes or nodes running PC host code. Combining the different levels in the same simulation gives both a fast simulation and fine-grained execution details on selected nodes. COOJA is also useful for executing firmware from multiple OSes in the same simulation. This is shown in Paper II and Paper III, where we perform interoperability testing in COOJA with both Contiki and TinyOS.
1.3 Research Methodology
The research methodology used during the work with this thesis is experimental computer science. The methodology is defined in the publication “Academic Careers for Experimental Computer Scientists and Engineers” \[11\] as “the building of, or the experimentation with or on, nontrivial hardware or software systems.” In my case, I have focused on developing nontrivial software systems for resource-constrained hardware combined with low-power wireless networking. In experimental computer science, we first set up a hypothesis on the expected behavior or performance of the system under experimentation. This hypothesis is then confirmed (or rejected) via evaluation of the system. Typically, the evaluation of these type of systems use metrics that focus on the combination of constraints and performance, for example, energy consumption, packet reception ratio, and memory consumption.

The experiments included in this thesis have been performed in simulation of such hardware and on real IoT devices in testbeds. One of the included results is also evaluated in real deployments with hundreds of devices.

1.4 Dissertation Structure
The dissertation contains two parts: Part I, a dissertation summary that introduce the research topic and summarize included publications, and Part II, the included research publications.

The first chapter in Part I introduces the research topics and the technology involved. Chapter 2 describes the relevant research challenges in the area. Chapter 3 discusses the contributions and impact of our research. Finally, Chapter 4 contains a summary of included publications and a description of my contribution.

The second part contains the five publications that are included in this dissertation.
2. Research Challenges

There are a wide variety of challenges within the research field of Internet of Things. This section covers the ones that have been my focus during the work with this dissertation. They are mainly related to low-power networking of constrained devices.

2.1 Power Profiling and Energy Efficiency

Most IoT devices are battery-powered and are often designed to have a battery lifetime of five to ten years [53]. This is the main reason for the use of low-power components such as small MCUs, low-power radios and resource-efficient operating systems and communication stacks. When comparing the energy consumption of different implementations, it is important to know the energy consumption in detail. Keeping track of the energy consumption can be done in software by measuring the on-time of each hardware component [16, 29], or by external measurement hardware that measures the accumulated power consumed by the device. One example of a hardware-supported power profiling testbed is the D-Cube benchmarking infrastructure [63, 64] which provides fine-grained measurement of power consumption in a controlled wireless environment. The research challenge in focus of the thesis is how to achieve energy consumption estimation of IoT software with low overhead and high accuracy. Paper I investigates this challenge.

2.2 IoT Network Interoperability

Interoperability between implementations of IoT network stacks is an essential feature for the commercial success of wireless IoT networking technology. Interoperability testing is fairly easy on higher levels of the protocol stack when two implementations can exchange some application messages and easily know what to expect. Testing a complete IoT mesh stack is a more complex task that requires many nodes running the implementation to form the mesh topology before the test can be completed and interoperability can be evaluated. Most of the interoperability tests are performed by bringing implementations and devices to interoperability test events and setting up a series of tests for evaluating various aspects of interoperability. The main challenge related
to interoperability is how to efficiently test multiple implementations in complex network topologies. Many tests also lack quality of the output, as they are focused on determining interoperability between implementations. There is a need for more quantitative metrics for evaluating performance when interoperability is achieved, e.g., performance when running multiple implementations. In simulation-based testing, it is possible to extract information about processor and memory usage, packet forwarding delays, and many other parameters from all nodes. This information is useful for calculating performance metrics for the protocol implementations under test.

The specific research challenge addressed in this thesis is how to perform interoperability tests of IoT software that are fast, automatic, and provide fine-grained data. Paper II and Paper III, explore this challenge.

2.3 IoT Network Scalability

IoT devices can communicate using various wireless technologies such as WiFi, Bluetooth, and 6LoWPAN. While working with Yanzi Networks, an IoT startup, we realized that gateway costs dominated the cost of the whole IoT system, both hardware cost and the deployment cost of the gateway. Therefore, the IoT network protocols must allow the IoT network to scale to a large number of devices. Bluetooth networks have typically been able to attach just a few devices per gateway, while most WiFi systems are able to handle hundreds of devices. Both these technologies use star network topologies by default. This means they cannot handle devices more than one network hop away from the gateway. This thesis focuses on 802.15.4 and 6LoWPAN, which allow more complex multi-hop network topologies. The challenge of scalability comes from the fact that the devices are resource-constrained. For example, they cannot store large number of neighbor entries and routes in the neighbor and routing table at the same time as they need to keep data about neighbors and links. This is because the cost of rediscovering that data is too high with respect to both energy and bandwidth. Routing devices in dense deployments will have lots of potential neighbors to store in the tables, and each routing device might need to route for many devices further away from the gateway.

Another issue with network scalability in most IoT networks is related to the wireless medium itself. In a high-density environment where many devices might transmit simultaneously, the reception of the data will likely fail due to packet collisions at the receiver. This can be managed in multiple ways (time slotting, channel hopping, etc), but when the density of devices is too high, it will still limit the scalability of the network.

The research challenge in focus of this thesis is how to support scalability of an IoT network given a high number of neighboring nodes, even when the IoT devices are significantly resource-constrained. Paper V addresses this challenge.
3. Contributions

The contributions that are the results of the work behind this dissertation are focused on the research challenges described in the previous chapter. Using our operating system and the simulation tools I have developed as a part of this thesis, I have shown that (1) it is possible to automate the process of and add quantitative metrics to interoperability testing, an important activity to achieve the success of standards-based IoT protocols and (2) by careful management of neighbor and routing information, it is possible to significantly improve the scalability of the RPL routing protocol.

Since experimental computer science not only results in specific research contributions but, in many cases, also provides the software and tools to repeat the experiments, we will report both types of contributions. In our case, this means that many of the software systems developed for the experiments are available as open-source software on GitHub, mainly in the Contiki-NG repository.

In the rest of this chapter, I will first elaborate on the research contributions, followed by a separate section summarizing software contributions.

3.1 Operating System and Simulation Support for Network-Scale Power Profiling

Measuring power consumption at a high resolution is very important when designing ultra-low power communication protocols and smart utilization of peripherals [41]. As Contiki is designed for low-power, many mechanisms for low-power operation and energy consumption estimation have been added. For example, Contiki contains an energy estimation module [16] that measures time in relevant energy states of the most important components, including MCU and radio. This will give each device an internal estimate of its energy consumption at a fairly low cost in terms of RAM, flash, and processor cycles on the device. In a simulated environment like COOJA/MSPSim this type of measurement can be performed by the simulator itself without any cost in terms of resources in the simulated devices. In simulation, the measurement can also be made more fine-grained as there is less limitation on the clock speed and resolution of the timers available to the IoT firmware. In Paper I, we show that COOJA/MSPSim based energy estimations correspond well with results using similar methods on real hardware, making it a powerful tool for comparing different implementations in simulation, even when the firmware does not support energy estimations.
3.2 Interoperability and Performance Testing for IoT Stacks

Interoperability and performance testing of wireless low-power protocols is typically done via test fest events or interop events where the main purpose is to produce interoperability test reports that consists of a matrix of answers on which stacks were interoperable and what features they are tested on [21]. Our contribution to improve interoperability testing is twofold: (1) we make use of simulation and emulation tools to speed up testing, and (2) we evaluate some performance metrics beyond those typically used.

In Paper II we show that we can run simulations consisting of nodes executing firmware based on different operating systems. In this specific case, we show both Tiny OS and Contiki running in the same simulation and communicating with the same protocols. We also evaluate the power consumption of both stacks via the built-in energy estimator in COOJA/MSPsim (described in Paper I), making it possible to evaluate the energy efficiency of communication stacks and IoT applications. The main contribution of Paper III is that it takes the interoperability and performance testing one step forward by testing the interoperability of two complete 6LoWPAN stacks, including the RPL routing protocol. This task previously required meeting up and running tests with complete software and hardware. Using our approach, the development team can iteratively improve the code base while the tests are run entirely emulated in COOJA/MSPSim. One of the most valuable features of simulation and emulation is the zero cost of full state inspection of both devices and all communication, making debugging very fast compared with physical interop tests. We found bugs and incompatibilities in both stacks during the work with Paper III.

3.3 Network Scalability under Memory Constraints

During my work with the IoT start-up Yanzi Networks, we did a series of scalability experiments by setting up hundreds of IoT devices and evaluated network topology stability and reliability of the network. The initial stability was far from satisfying due to intensive neighbor churn. Since we use storing mode of RPL, which stores routes to the next hop in each node, the neighbor churn caused inconsistencies in the topology. In the specific hardware used, STM32W, it is impossible to increase the neighbor table to fit all neighbors seen, nor handle all route requests received. In total, this device has 16 kB RAM and the tables allowed ten neighbors and 20 routes. Paper V’s contributions are (1) an end-to-end route registration mechanism for storing mode RPL, so that registering nodes know if the route was registered all the way to the gateway, and (2) a neighbor policy mechanism that protects neighbors from deletion and carefully controls which new neighbors enter the neighbor table.
These two mechanisms solved the scalability issue for the RPL routing protocol for the Yanzi devices, and we can now scale beyond 600 devices, all within single-hop communication range, with only tens of neighbors and routes in the tables. We have upstreamed this improvement to Contiki’s GitHub repository.

3.4 Software Contributions

During the research work I have been doing, I have developed software that, most of the time, ended up as open-source contributions. I am the initiator and leading developer of the MSPSim emulator and its integration into COOJA. MSPSim has provided our research team and many other research teams with a very efficient tool for debugging, measuring, and evaluating new code and mechanisms for IoT, including low-power protocol implementations. MSPSim has been extended and used in many research projects. Some examples are: SIREN, that extends MSPSim for batteryless energy harvesting devices [23], Shimmer that adds support for Bluetooth communication [42], and Stecklina et al.’s work that implements support for secure memory models [66].

I am also one of the two leading developers of ContikiRPL, which our research team and many others in academia have used for their research [6, 18, 71]. Our RPL implementation is the basis for the default routing protocol in Contiki-NG, and we were among the first to implement RPL during the standardization process within the IETF ROLL-working group.

Furthermore, I was the lead developer of our LwM2M stack and the IPSO-objects implementation in Contiki. This implementation has been part of interop tests, used by commercial companies, and integrated into Zephyr OS when they needed an LwM2M implementation.

Finally, before starting my research on sensor networks and IoT, I was part of the research team that arranged the AI-competition Trading Agent Competition, TAC, for several years. I was the lead developer of the implementation of the competition servers during both TAC Classic and TAC SCM - a game that we co-developed with Professor Norman Sadeh and Raghu Arunachalam at Carnegie Mellon University. We arranged the competition at several AI conferences, including AAAI. Our publication for TAC-03 [60] has more than a hundred citations, and several hundred publications are related to the competition.
4. Summary of the Papers

4.1 Paper I: Accurate, Network-Scale Power Profiling for Sensor Network Simulators


Summary
In this paper, we evaluate the accuracy of the combined sensor network simulation tool COOJA/MSPSim, which consists of COOJA, a sensor network simulator, and MSPSim, a sensor node emulator. The evaluation uses Contiki’s power profiler as baseline [16]. The power profiler measures time spent in different modes for each chip on a node and calculates power consumption by multiplying time with a pre-measured current draw and battery voltage. We compare experimental results measured on real sensor nodes with simulation results for three different MAC protocols. The MAC protocols are of varying types, one is TDMA-based (CoReDac), and one is low-power probing (LPP), and the final one is based on low-power listening (X-MAC). The results of the evaluation indicate that COOJA/MSPSim enables accurate network-scale simulation of the power consumption of sensor networks.

Contribution
The main contribution of this paper is that we evaluate the accuracy of power profiling in simulation by comparing the results from the simulation with results from execution on real sensor nodes. We did this evaluation on a network scale, which differs from previous efforts that only evaluated single nodes without any communication aspects. Another important contribution is the simulation tool, COOJA/MSPSim that supports accurate power profiling.

My Contribution
I am the main developer of MSPSim, and during the work on this paper, I improved it for better support of power profiling, improved the CC2420 radio chip emulation, and extended the integration with COOJA. I also made some of the experiments and wrote parts of the paper.
Reflections
In this paper, we evaluated and improved the simulation’s power profiling features, a vital part of speeding up research and development of energy-efficient software and communication protocols for IoT. More than 50 other publications have cited the paper, and this type of feature is still crucial since energy efficiency is still an important feature of IoT and other systems.


Joakim Eriksson, Fredrik Österlind, Niclas Finne, Nicolas Tsiftes, Adam Dunkels, Thiemo Voigt, Robert Sauter, Pedro José Marrón.

Summary
In this paper, we show that COOJA/MSPSim can be used for interoperability tests between different protocol stack implementations in different sensor network operating systems. We also, show that the built-in power profiling in MSPSim is as accurate as the Contiki’s power profiler and that it can be used for power profiling any application without any power profiling support from the operating system in the node. We evaluate COOJA/MSPSim for use in interoperability tests by adding support for TinyOS and performing basic experiments where nodes based on TinyOS communicate with nodes based on Contiki OS.

Contribution
The main contributions of this paper are that we show that COOJA/MSPSim can be used as an interoperability testing tool and that it accurately evaluates the power consumption of the simulated nodes. Interoperability testing in COOJA/MSPSim gives the tester much more detailed information than performing the same test on real nodes since the full state in all nodes can be inspected at any time.

My Contribution
I performed some of the research experiments and wrote parts of the paper. I also improved MSPSim for better support of the radio chip CC2420, specifically to meet the needs of TinyOS such as support for SFD capture interrupt.

Reflections
This is the first paper where we show interoperability testing in simulation tools. At the time of writing, this work has 290 citations and is still cited in new
publications in 2023. New full system simulation tools like Renode [7] can add another aspect of heterogeneity regarding hardware variations. Integrating this with IoT wireless network simulators like COOJA is an exciting opportunity, allowing even more variety in the testing scenarios.

4.3 Paper III: Industry: Beyond Interoperability - Pushing the Performance of Sensor Network IP Stacks

JeongGil Ko, Andreas Terzis, Johns Hopiks University, Joakim Eriksson, Nicholas Tsiftes, Adam Dunkels, Stephen Dawson-Haggerty, David Culler, Jean-Philippe Vasseur, Mathilde Durvy In proceeding of SenSys 2011, November 1-4, 2011, Seattle, WA, USA.

Summary
In this paper, we present two interoperable implementations of the IPv6 protocol stack for low-power and lossy networks. The stacks also use two different operating systems, Contiki OS and TinyOS, representing two independently developed stacks. The interoperability between the stacks is evaluated on a basic IP-packet exchange and validation of the behavior and network performance when mixing a varying percentage of the two stacks. We assess the cost of interoperability and show that the performance and overhead are similar to more specialized state-of-the-art low-power protocol stacks. We evaluate performance in sensor network testbeds and COOJA/MSPSim, simulating complete networks with both stacks.

Contribution
The main contribution of the paper is that we evaluate interoperability with respect to additional performance metrics compared to other efforts such as plugfests [56]. We also show that performance depends on the topology and the number of nodes running each stack. Finally, we also illustrate the usefulness of using simulation tools to trigger hard-to-find bugs and interoperability issues.

My Contribution
I worked mainly with the simulation experiments and finding and fixing bugs and issues in the ContikiRPL implementation. I also found and reported some issues with the TinyOS implementation. I did parts of the evaluation and wrote parts of the paper.

Reflections
This is a follow-up paper on interoperability tests in simulation. We extend the work from the previous with a collaboration with the TinyOS team and
evaluated not only interoperability as a binary statement but also added other performance metrics. Given the recent success of the 6LoWPAN-based protocol Thread [68], this type of testing is getting even more relevant and might, in the long run, enable testing of both interoperability and other metrics, including energy consumption ratings in the same tools. This would allow for similar labeling as household appliances’ energy ratings - such as EU’s energy consumption labeling scheme.

4.4 Paper IV: The Contiki-NG Open Source Operating System for Next Generation IoT Devices


**Summary**

In this paper, we present the Contiki OS fork, Contiki-NG. The first release of Contiki-NG introduces many new features, but it also re-uses, with or without modification, many of the features of the original Contiki OS, such as the scheduler, the event-based kernel, data structure manipulation libraries and storage. Contiki-NG also uses, with minor modifications, multiple networking-related software components, including the original implementations of 6LoWPAN and RPL (called RPL Classic in Contiki-NG). For improved scalability, Contiki-NG provides RPL Lite, a cut-down version of the RPL implementation that only supports the non-storing mode of RPL but is more lightweight. The paper also describes the new focus on reliability and security and the process used for releases and pull requests to the Contiki-NG GitHub repository.

**Contribution**

The paper is the first publication that describes Contiki-NG and the purpose of forking. While not adding any new research in itself, the paper summarizes many of the efforts of the work that have resulted in a solid support for IP-based communication, 6LoWPAN, energy-efficient communication, and a toolset for research and development of advanced IoT applications.

**My Contribution**

I was one of the leading people who set up the Contiki-NG repository and worked on the cleanup and new ways of working with the codebase. I was part of the team that wrote the paper. I wrote the LwM2M section of the paper as I am the main developer of the LwM2M implementation in Contiki-NG.

**Reflections**

As mentioned above, this paper is not a scientific publication and does not add any new research, but is included in my thesis as it represents the re-
results of many years of research that have been performed using Contiki and Contiki-NG. This paper contains the software-based results from many of my additional publications that are not part of the thesis and a whole body of work by others. This paper is the first Contiki-NG publication from the Contiki-NG team, and it has more than 60 citations at the time of writing.

4.5 Paper V: Scaling RPL to Dense and Large Networks with Constrained Memory


Summary
In this publication, we show that the routing protocol RPL can handle very dense and large-scale networks even with small routing and neighbor tables. We add a new standard-compliant mechanism to RPL that details some parts of the RPL RFC that ensures that a node in the network that requires downward routing will know that it has a route from the gateway. We also show that this mechanism, end-to-end DAO, improves the network’s reliability. We also added a neighbor table policy that protects neighbors used as a next-hop in routing from being removed. The neighbor policy is designed not to require a large neighbor table even if the network topology is dense and significantly more neighbors exist than can be stored in the table. Both mechanisms are designed to avoid conflicts with the RPL RFC to enable interoperability with nodes not implementing these mechanisms. We evaluate both mechanisms and show that they enable stable RPL topologies even when nodes cannot store all neighbors in the tables.

Contribution
The main contribution of the paper is the design and evaluation of the two mechanisms that allow storing-mode RPL to scale well in both large and dense networks without extending the RAM of the routing devices.

My Contribution
I am the main designer and implementer of both mechanisms in the ContikiRPL routing implementation. I have also performed research experiments and written most parts of the paper.

Reflections
Both issues with storing-mode RPL did get attention in the IETF Roll working group, and at least one informal RFC per issue has been active since the publication of the paper [33, 34]. Today, most of the current applications using
RPL do use the non-storing mode that scales better in itself and has the end-to-end DAO-ACK built-in. However, even non-storing mode will need a strict neighbor management policy to handle dense networks with more neighbors than can be stored in the neighbor table. Otherwise, nodes might have to drop packets during forwarding when the next-hop neighbor is not in the table.
5. Related Work

This chapter discusses related work on the research topics covered in this thesis. It also includes some related work on operating systems and simulation tools, which have been essential for most of this thesis’s research.

5.1 Operating Systems

Most of this thesis’s research work heavily relies on embedded operating systems. This section describes related operating systems for IoT applications.

TinyOS [28] is one of the first operating systems for IoT and wireless sensor networks. Similar to Contiki, it is an event-driven OS designed for resource-constrained hardware such as the Tmote Sky. Tiny OS made use of a component-oriented C-language extension called nesC and a compiler that produced a compact C-program for compilation with a regular C compiler. Learning this new language requires effort from developers, which is a significant difference from Contiki, where the OS is developed in plain C. While very successful in academia, TinyOS never established itself as a commercial product OS.

RIOT OS is an IoT OS designed with a focus on resource-constrained devices [8]. While Contiki is an event-driven OS, RIOT is a fully multi-threaded OS with real-time support. There is support for many of the latest IETF standards related to IoT, including 6LoWPAN, RPL and CoAP. Unlike most other IoT operating systems, RIOT decided on a GNU LGPL license, which might somewhat limit its usage in commercial products, but there is academic support for RIOT.

Tock [46] is an operating system for low-power resource-constrained platforms. Tock focuses on software security and uses the hardware protection mechanisms and type safety of the Rust programming language [13]. Tock isolates software faults, provides memory protection, and manages memory for dynamic applications. A key difference between many other IoT OSes and Tock is the strong isolation of applications that allow concurrent execution of applications in similar ways as larger OS-es (Linux, Windows, etc). Currently, Tock is available on a large number of platforms and illustrates how to achieve software security using modern MCU architectures combined with a modern programming language. While Contiki focuses on communication, Tock focuses on a secure software runtime and lacks communication stacks in the OS source code.
Zephyr OS is another operating system for resource-constrained IoT devices. Zephyr is hosted under the Linux Foundation but makes use of the Apache license to allow usage without forcing the release of source code. Zephyr OS has very strong platform support with hundreds of different boards supported. The operating system originates from a commercial OS called Virtuoso, but it was open-sourced in 2016. Initially, Zephyr borrowed the 6LoWPAN and RPL implementation from Contiki but re-implemented the complete IP stack since then. Zephyr supports IPv4/IPv6 and BLE but lacks support for 6LoWPAN IoT low-power mesh with RPL and TSCH (6TiSCH). Zephyr OS has the strongest industrial support with companies including Intel, NXP, and Nordic Semiconductor as a part of the project’s consortium.

Recently WebAssembly [70] (WASM), a virtual machine initially designed for safe and fast execution of software modules in the browser, has been used for IoT devices. The WASM ecosystem includes several open-source implementations of the runtime and a large number of compilers and tools for building software modules from a wide variety of programming languages. Currently, there are no complete IoT operating systems fully based on WASM, but there are examples of IoT systems using WASM. One example is the WASM implementation WAIT [47], which shows that WASM can be used on resource-constrained IoT devices and provides a set of IoT-related APIs for WASM-based software modules.

5.2 Simulation Tools

Simulation tools are essential for evaluating and testing during the development of new IoT communication protocols and software features. Many of the simulators are for a specific IoT OS, but some are also generic IoT and networking simulators.

TinyOS simulation tool, TOSSIM [45], allows the simulation of complete TinyOS systems, including communication between simulated sensor nodes. TOSSIM, TinyOS’s simulation tool, compiles to the native CPU architecture of the machine that is running the simulation (typically x86). Unlike the COOJA simulation platform, TOSSIM cannot emulate IoT devices.

Renode [20] is an emulation and simulation platform for modern processor architectures such as ARM, RISC-V and Intel x86. Renode supports multi-device simulations and supports radio medium and wired connectivity between devices. The focus of Renode is to enable continuous integration tests of complex embedded hardware and software setups. Renode lacks support for some of the ultra-low-power platforms that made COOJA successful (msp430 / Tmote Sky) in the wireless sensor network research field but supports modern replacements such as ARM Cortex M3 and SoC’s based on that MCU architecture.
The widely used Network Simulator, NS, currently at version 3, is a discrete event network simulator developed in C++ and Python. NS-3 is more of a simulation building library than a complete simulation application with GUI like COOJA. Simulations in NS-3 are defined in C++ and scripted via Python scripts, a significant difference from COOJA where configuration files describe the simulations. As NS-3 mainly focuses on IP networking over ethernet and WiFi it lacks many of the protocols and features of the IoT devices supported in COOJA.

IoTSim-Edge is a simulation framework that models both IoT devices and Edge platforms in the same simulator [35]. IoTSim-Edge supports a set of IoT protocols and models both the performance and energy consumption of supported protocols. It also provides a way to model edge applications and their analytics operations and modeling of mobile IoT devices and the handover of mobile devices between edge nodes. A significant difference between COOJA and IoTSim-Edge is that the latter does not include emulation or OS-level native execution. Thus all modeling is on a higher level which means it will not model all details of the IoT devices’ software and behavior.

IoTNetSim is a simulator for end-to-end modeling and simulation of IoT systems, including IoT devices, the network with network devices and the cloud [61]. IoT, fog, and edge nodes can be configured for connection type to model communication speed correctly. IoTNetSim models the complete IoT network, data collection, energy consumption, and analytics in the cloud. IoTNetSim can predict battery lifetime, the storage size of data collected, and many other parameters. The level of abstraction is much higher in IoTNetSim when compared to COOJA. COOJA uses the actual implementation of the application and communication protocols, while IoTNetSim models the application and protocols. COOJA is also limited to simulating the low-power IoT network, while IoTNetSim models the whole end-to-end IoT system and application.

Kaala is another simulator designed for end-to-end Internet of Things (IoT) systems [12], aiming to enhance the realism of IoT simulations by integrating with cloud services from providers such as Azure, AWS, and Google. Kaala provides emulation of IoT devices by running application logic in separate processes. Unlike COOJA/MSPSim’s emulation, where low-power IoT devices are emulated and run their original firmware, Kaala uses docker and Mininet [38] to set up Linux-based emulation IoT devices in a realistic network topology. Kaala’s emulation is more similar to COOJA’s simulation when nodes are executed on the native host, but the focus on Kaala is more on the whole IoT system, including IoT cloud services, while COOJA’s focus is on low-power IoT network simulations.
5.3 Energy Estimation

Energy estimation support, either in software or hardware, is a critical feature for research on energy-efficient IoT communication and applications. This section covers work related to energy estimation. Typically, an energy estimator will measure the energy used for some important components. Within IoT, the microprocessor and the radio are the key components that need to be measured to understand the impact of computation versus communication.

PowerTOSSIM [65] is an extension to TOSSIM [45], TinyOS simulation tool, for estimating per-node power consumption. However, TOSSIM and PowerTOSSIM only support simulations of nodes at the operating system level, while COOJA supports detailed emulation on some platforms. Emulation will typically give a better estimation as the instruction sets in the different MCU architectures and differences in compilers can cause a difference in timing on the code-block level.

AEON [43, 44] is a tool for predicting the power consumption of TinyOS applications using the AVRORA emulator. AEON can predict TinyOS applications significantly better than PowerTOSSIM and is a similar approach to the one taken in COOJA and MSPSim where an MCU emulator is used to allow the predictor to access very detailed timing of all executed code and to track, in detail, the usage of all different hardware components. The main difference between COOJA and AEON is that COOJA supports all sorts of simulation/emulation combinations (e.g., both OS-level simulation and emulator-driven simulation) while AEON is only emulator-driven. AEON also only supports Avr-based platforms, while COOJA supports MSP430-based platforms.

Jha et al. [35] present IoTSim-Edge, a simulation platform that focuses on the ability to evaluate the performance of heterogeneous IoT devices and edge configurations. It simulates the behavior of communication protocols and applications rather than emulating the devices as COOJA/MSPSim does. IoTSim-Edge provides models for the energy consumption of the devices and the communication. Still, since it is a simulation, it will not evaluate the actual applications and protocol implementations intended to be run in the various devices but an estimate based on simulated protocols and applications.

Another option for software-based energy estimation is to use a complex testbed with energy measurements of the device while measuring GPIO pins that relate to different states of the radio, microprocessor, and other components. This can at times be somewhat less intrusive as software-based estimators need to add code for tracing the component usage. However, even with a hardware-based energy meter, the software needs to be modified to indicate the type of mode the microprocessor and the radio are in. Flocklab [48] uses this approach and has a sophisticated hardware solution for high-resolution measurement of power consumption. While less intrusive on devices like Tmote Sky, where the radio is separate from the MCU and therefore possible to moni-
tor via hardware, it is not as non-intrusive on modern SoC-based devices where the radio and MCU are in the same chip.

One of the most sophisticated evaluation frameworks for wireless IoT networking is D-cube [63, 64]. It extends the metrics beyond energy-related properties and adds end-to-end latency and a set of reliability metrics for fully benchmarking IoT networking. D-cube is mainly a software infrastructure and supports multiple hardware devices as IoT devices under evaluation, and it uses dedicated hardware for high-quality energy consumption estimations. For the highest possible quality of the measurements, D-cube requires some minor modifications of the evaluated firmware (for precise timing indications, etc).

In many cases, energy estimation focuses on estimating the lifetime of battery-powered devices. Quite often, an ideal battery model is used. Dron et al. [14] add a battery model into the calculations to better estimate the device lifetime. The improved lifetime estimator uses COOJA and MSPSim to allow detailed simulation of the whole IoT network, and they show that the ideal battery model commonly used overestimates network lifetime by more than 30 percent.

5.4 Automated Interoperability Testing

IoT and its communication protocols started with proprietary protocols and research experiments but is now a mature field with many standardized protocols and commercial products. Interoperability testing is a critical tool for developing standards and commercial products, and automation of interoperability testing can significantly improve development efficiency.

Rosenkranz et al. [59] present a distributed testing platform for interoperability tests. They claim that emulation and simulation-based interoperability testing allow white-box testing, but it is not feasible when the number of emulated devices is too high. They propose a distributed testing framework that allows platform providers to plug-in to the test and even connect parts of the framework remotely. This enables what the authors call a permanently available, distributed plugtest. This allows efficient testing, but there are limitations when it comes to timing. A platform with these limitations is likely not useful for testing time-synchronized protocols (e.g., TSCH) that require millisecond precision of timing possible in emulated environments such as COOJA/MSPSim or even with a detailed simulation as with COOJA’s native simulation.

PatrIoT is another IoT interoperability and integration testing platform with the ability to perform tests in simulation or physical testbeds [10]. The main focus of PatrIoT is testing the interoperability and integration of complete IoT cloud services using simulated IoT devices, which differs from our approach, where we use COOJA/MSPSim for testing the IoT protocol implementations for interoperability.
5.5 Scalability of IP-based IoT Mesh Networks

Iova et al. [32] identify RPL Scalability as an issue where RPL implementations in Contiki and TinyOS illustrate that RPL is not scalable on low-power devices due to high memory requirements w.r.t both flash and RAM. The authors evaluate storing mode RPL and conclude that RPL is not scalable on low-power devices as they will quickly hit the limit of neighbors and routes the devices can store. The evaluation makes use of smart city topologies that consist of very few paths to the root node and might cause problems. In dense deployments with many paths to root it is, even using storing mode, possible to scale with a reasonably stable topology given a good neighbor policy and using end-to-end DAO registration as we have shown in the final paper in this thesis.

Thread [25] is another 6LoWPAN-based IoT mesh protocol with roots from NEST’s smart thermostat and smart home mesh networks. Thread is a low-power mesh network standard, based on distance vector routing. Thread allows up to 32 simultaneously active routers. Child nodes, or non-router nodes under each router, are assigned an IPv6 address indicating which router they are associated with. Thread has a limited scalability per sub-network, but recently, it has been improved to handle routing between multiple sub-networks. According to the Thread Group, the Thread protocol scales to a few hundred nodes. RPL, on the other hand, is capable of handling large networks but the routing model is significantly different as it creates a DODAG (destination-oriented directed acyclic graph) that is typically less dense than the Thread mesh. This routing model often results in a longer path between two pairs of nodes. If needed, it is possible to improve the path length issue in RPL-based Networks either by adding multiple gateways or using specific RPL protocol adaptations.

There is a set of other alternatives for IoT connectivity that are not IP-based mesh but are still relevant alternatives for some applications. LoRa is one of the long-range alternatives that allow for low-power and long-range communication. LoRa nodes do not communicate with each other directly. Instead, they are configured as star networks, with a central gateway that relays messages between the nodes. Telecom networks are also starting to provide IoT-related connectivity with NB-IoT [58] and other 5G mechanisms for low-power connectivity. Unlike other solutions, these will have a dedicated spectrum, allowing careful network planning without interference from other networks. Finally, Wi-Fi 6 release 2 [5] provides new energy management features for low-power IoT devices, making Wi-Fi a better option for IoT than previously. Most of these other mechanisms are based on star networks, which is good for the ability of the device to keep a low memory usage for networking and also allow longer sleep times as the gateway is typically always awake. The main issue with star networks is range since all devices must reach the gateway in one hop.
6. Conclusions and Future Work

This chapter concludes the thesis and describes some potential future work.

6.1 Conclusions

This thesis presents research contributions within the wireless sensor network and Internet of Things research areas. The contributions focus on communication and networking with low-power and resource-constrained devices.

Paper I focuses on enabling energy estimation at network scale. We show that it is possible to estimate power consumption even when there is no support in the firmware using a time-keeping mechanism that records usage time per hardware component and their state in a simulated environment. This enables instant comparison of low-power protocol implementations and other low-power mechanisms.

Paper II and Paper III demonstrate that interoperability testing can be more than a large physical meeting where lots of protocol implementations are tested against each other, with a binary outcome. We show that we can perform the interoperability tests in simulation and that it is possible to achieve continuous testing as the stack implementations are always available for testing in the simulation environment. We also show that interoperability tests using fine-grained simulation can provide interesting information in terms of performance metrics that can guide further development and give more detailed understanding of interoperability in complex environments (such as multi-hop mesh networks). During this work, we used our simulation platform COOJA/MSPSim, but in the future, we will likely use tools like Renode that supports more modern system-on-chips.

Paper IV presents Contiki-NG, a new Contiki fork created to establish a clean version of the OS for modern IoT system-on-chips, focusing on standards-based protocol implementations. Contiki and Contiki-NG represent the main software stacks I have used for this thesis’s research. The Contiki-NG repository is also where most of the code from my research is published.

Finally, Paper V describes our results relating to network scalability and topology stability when deploying large and dense IoT networks. This work takes its challenge from real-world deployments. We design and evaluate several mechanisms and show that it is possible to improve scalability by carefully managing neighbors and ensuring end-to-end information about routing topology. This work also resulted in input to IETF’s ROLL working group and an IETF draft on neighbor management policy in the IETF LWIG working group.
Using our operating system and the simulation tools I have developed as a part of this thesis, I have shown that (1) it is possible to automate the process of and add quantitative metrics to interoperability testing, an important activity to achieve the success of standards-based IoT protocols and (2) by careful management of neighbor and routing information, it is possible to significantly improve the scalability of the RPL routing protocol. I believe that these results are applicable in other IoT and low-power networking scenarios.

6.2 Future Work

In the near future, we will continue improving Contiki-NG and its 6LoWPAN stack, especially for ultra-low power devices. Contiki-NG is soon too small and limited to be relevant in today’s common IoT System-on-chips (ARM-32 bit with a significant amount of RAM, typically 64 kB or more), at least if the IoT applications continue to be low-complexity sense-and-communicate applications. However, the current trends of integrating machine learning (ML) into IoT devices might change things to focus on resource efficiency on the OS and application level to allow more resources for machine learning. Future work for our research related to Contiki will include supporting machine learning mechanisms and adding secure and lightweight execution environments. One of the secure execution environments in focus right now is WebAssembly [50, 70]. It is currently gaining much attention and is a perfect fit for IoT devices due to its low resource requirements. Combining the two trends of machine learning in IoT devices with support for both ML models in the devices and the needed software logic for the ML models is an exciting opportunity. There is already some ongoing work in this direction, like WebAssembly System Interface (WASI) and WASI-NN neural network interfaces for WebAssembly. With support for this, it would be easier to deploy ML-models and the supporting software in any IoT device that supports WebAssembly and WASI-NN, independent of their underlying hardware. This will likely be an essential part of the future for IoT software within a few years.

We will also look into what type of impact the private 4G/5G network for industries, buildings, and cities will have on low-power IoT networks. These 4G/5G networks allow any industry or city to get their spectrum for hosting a wireless enterprise network based on telecom technologies, using the complete system from telecom providers or only using the hardware (e.g., base stations) and combine that with a fully open-source core network [27]). Combining local low-energy IP-based mesh networks with these long-range networks 4G/5G or even LoRa is an exciting research area for end-to-end, low-power, and secure connectivity for large-scale IoT deployments.

Finally, an interesting area where a tiny and energy-efficient operating system is relevant is the battery-free devices such as the Onio.zero [54]. The
resource constraints of the devices are similar to what was the case when the development of Contiki started more than fifteen years ago. The Onio.zero has 2 kB RAM, up to 32 kB flash, and a 32-bit RISC-V MCU and can use energy from radio transmitters, light, and other external energy sources. Challenges include how to manage computation and networking when having intermittent power and how to achieve secure and standards-based communications. This is something we will also likely investigate.

Under den forskning som ligger till grund för avhandlingen har jag använt och vidareutvecklat Contiki och Contiki-NG som är operativsystem för IoT-enheter. Jag har även använt och vidareutvecklat simuleringsverktyget COOJA/MSPSim som har förmågan att simulerar kompleta IoT-nätverk där radiomediet och alla IoT-enheter simuleras. COOJA/MSPSim stödjer detaljerad simulering av IoT-noder och möjliggör därmed detaljerad analys av kommunikationsprotokoll och dess egenskaper.

För att kunna skapa energieffektiv mjukvara för IoT-enheter behövs verktyg för att utvärdera energiförbrukning för hela IoT-nätverk. I vårt simuleringsverktyg COOJA/MSPSim finns stöd för att uppskatta energiförbrukning på ett detaljerat sätt genom att mäta hur lång tid olika delkomponenter som radio, mikroprocessor och sensorer är aktiva. COOJA/MSPSim kombinerar simuleringsverktyg med en detaljerad emulering av modernas
mikroprocessorer vilket ger en verklighetstrogen och högupplöst uppskattning av tidsåtgång för varje simulerad nod. Vi visar att det går att få både detaljerade och verklighetstrogna resultat för hela det simulerade IoT-nätverket.

En annan möjlighet är att använda simuleringsramverket för att testa och utvärdera olika implementationer av IoT-protokoll. I detta sammanhang är fokus på att avgöra om de olika implementationerna är interoperabla, dvs. att de kan kommunicera med varandra. I många fall utförs interoperabilitetstester vid fysiska möten där alla tar med sig sina respektive implementationer och utför tester i en fysisk testmiljö. Vi visar att det går att snabba upp interoperabilitetstester genom att använda simuleringsbaserade tester av multipla implementationer. I och med att vi kan få fram en hel del extra information under simulering så är det möjligt att utvärdera mer än bara om de olika implementationerna fungerar tillsammans. Man kan bland annat få ut detaljer kring energieffektivitet och kvalitet på kommunikationen i nätverket, i olika storlekar av nätverket, densitet och blandning av de olika implementationerna.


Utöver de artiklar som inkluderas i denna avhandling så har vi även arbetat med att sprida resultat via bidrag till standardisering av IoT-routingprotokoll (IETF RPL) samt arbetat med att släppa mjukvara som öppen källkod. De flesta av de utvecklade mjukvarukomponenterna och simuleringsverktygen har blivit tillgängliga som öppen källkod inom ekosystemen för Contiki och Contiki-NG. Mjukvaran används av både forskare inom IoT och sensornätverk så väl som inom kommersiella IoT-projekt och produkter.
References


[48] Roman Lim, Federico Ferrari, Marco Zimmerling, Christoph Walser, Philipp Sommer, and Jan Beutel. Flocklab: A testbed for distributed, synchronized tracing and profiling of wireless embedded systems. In Proceedings of the 12th


Maria Salama, Yehia Elkhhatib, and Gordon Blair. Iotnetsim: A modelling and simulation platform for end-to-end iot services and networking. In *Proceedings


A doctoral dissertation from the Faculty of Science and Technology, Uppsala University, is usually a summary of a number of papers. A few copies of the complete dissertation are kept at major Swedish research libraries, while the summary alone is distributed internationally through the series Digital Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology. (Prior to January, 2005, the series was published under the title “Comprehensive Summaries of Uppsala Dissertations from the Faculty of Science and Technology”.)