Idle Listening Reduction Mechanism for Overprovisioned Cells in 6TiSCH Tracks

Mathias Utgård

Thesis submitted for the degree of
Master in Informatics: Programming and networks
60 credits

Department of Technology Systems
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2019

Idle Listening Reduction Mechanism for Overprovisioned Cells in 6TiSCH Tracks

Mathias Utgård

Abstract

The future of the (Industrial) Internet-of-Things (IIoT/IoT) is dependent on deterministic and reliable low-power wireless communication systems. Because of the limited battery capacity of devices within such a system, energy consumption must be kept to a minimum in order to extend the lifetime of a device.

6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4) enables deterministic low-power IP-enabled networking for industrial applications by leveraging the reliability of the Time Slotted Channel Hopping mode of IEEE 802.15.4-2015. 6TiSCH also introduces the concept of tracks as a mechanism to ensure end-to-end determinism within nodes of a 6TiSCH network.

In order to meet application requirements, tracks can feature overprovisioned resources in order to meet application requirements during "worst-case" scenarios. However, overprovisioning leads to unused track resources in the form of idle listening, which represents unnecessary energy waste.

This master thesis introduces and evaluates *Track Resource Adaptation* (TRA) as a mechanism intended to reduce the amount of idle listening events occuring in 6TiCH tracks.

Contents

List of Figures

List of Tables

Preface

This thesis concludes my master’s degree for the course of Informatics: Programming and Networks at the University of Oslo, Department of Informatics.

I would like to thank my supervisor Professor Knut Øvsthus and PhD candidate Andreas Urke at the Western Norway University of Applied Sciences for their invaluable insight and support.

Chapter 1

Introduction

Low-power wireless mesh networks are an important part of the future of digital communication. Concepts such as the Internet-of-Things (IoT) and the Industrial Internet-of-Things (IIoT) can be effectively realised by ubiquitous low-cost wireless sensor- and actuator devices and the technology and standards that surround them.

Advancements within Wireless Sensor Network (WSN) technology enable affordable "peel-and-stick" deployments of low-power wireless devices that can offer 99.999% end-to-end reliability for critical industrial applications [1]. These advancements have previously been available through WSN technologies such as WirelessHART [2] and ISA100.11a [3], which are tried and true standards for industrial wireless networking.

Currently, the Time Slotted Channel Hopping (TSCH) mode of IEEE 802.15.4-2015[4] is positioning itself as a very viable technology for WSN and IoT. Hence, it has been subject lot of attention within the research community, and is the key technology behind the standardisation efforts of the IETF 6TiSCH (IPv6 over the TSCH mode of IEEE 802.15.4) Working Group [5]. The aim of 6TiSCH [6] is creating a standard that enables deterministic IP-enabled networking for industrial applications. For this purpose, 6TiSCH adopts some concepts from *Determinstic Networking* (DetNet) [7]. This includes the concept of a DetNet *Deterministic Path*, referred to in 6TiSCH as a *track*. A track is a series of TSCH cells that form a Layer 2 path from a source to a destination, enabling determinism and reliability to traffic assigned to the track.

In order to meet application demands for latency and reliability, 6TiSCH tracks can feature overprovisioned TSCH cells. Overprovisioning provides extra cells for retransmission of packets, and also allows the track to better handle traffic peaks. Hence, overprovisioning is often used to allow an application to adhere to strict requirements for timeliness and reliability in a "worst-case" scenario. However, overprovisioning also introduces idle listening, where unused track cells at the transmitter causes the receiver to unnecessarily activate its radio, wasting energy. As WSN nodes usually run on batteries with very limited energy capacity, the amount of idle listening should ideally be kept as low as possible in order to extend the lifetime of the network.

This master thesis introduces *Track Resource Adaptation* (TRA) as a proposed mechanism intended to reduce the amount of idle listening events occuring in 6TiSCH tracks featuring overprovisioning. TRA gives nodes in a 6TiSCH network the ability to disable overprovisioned track cells in order to prevent idle listening when no traffic is expected. This is done by utilizing the pending bit of received packets as a decision maker for enabling or disabling cells. Later in this thesis, a more detailed description of TRA is given, and the results of simulations featuring TRA running in various scenarios is presented and analysed. Using simulation as a tool, the thesis will investigate the effect of TRA on metrics such as packet latency, packet loss, idle listening reduction and increase in network lifetime.

Related work

The following papers present ideas that are similar to this thesis' usage of the pending bit as signaling for toggling track cells.

Fafoutis et. al. [8] introduces Adaptive Static Scheduling, which aims at improving energy efficiency in static TSCH schedules featuring overallocation. This thesis' proposal of TRA is partly based on extending the idea presented in this paper into a 6TiSCH network.

Jin et. al [9] presents the AMUS scheduling algorithm, and discusses adding an End-Of-Queue notification to headers of transmitted packets in order to avoid unnecessary communication in the following cells.

1.1 Thesis Structure

The thesis is organized as follows:

- Chapter 1 (this chapter) gives the background and purpose of the thesis.
- Chapter 2 provides a collection of necessary theory related to the thesis.
- Chapter 3 introduces Track Resource Adaptation.
- Chapter 4 gives a description of the chosen method of evaulation.
- Chapter 5 details how the 6TiSCH simulator was used for this project.
- Chapter 6 presents the simulation scenarios considered.
- Chapter 7 contains the simulation results along with analysis.
- Chapter 8 draws a final conclusion of the work done in the thesis.
- Chapter 9 discusses eventual further work related to the project.

Chapter 2

Theory

This chapter will present some necessary theory related to the concepts surrounding the thesis. First off, a brief introduction to overarching topics such as the Industrial Internet-of-Things and wireless sensor networks is given, followed by a delve into the specifics surrounding the IEEE 802.15.4 wireless standard, Time Slotted Channel Hopping (TSCH), 6TiSCH, tracks, and the thesis proposal of Track Resource Adaptation.

2.1 The Industrial Internet-of-Things

Industrial Internet-of-Things (IIoT) refers to a trend within industrial automation where *Internet-of-Things* (IoT) technology is used to interconnect industrial devices. Primarily enabled by the convergence of industrial and IP-enabled low-power wireless networks, IIoT can broadly be thought of as the "next step up" from traditional Industrial Automation and Control Systems (IACS), and is considered an very important part of future industrial automation paradigms such as Industry 4.0 [10].

2.1.1 Definition of IIoT

[11] defines IIoT as:

> "A system comprising networked smart objects, cyber-physical assets, associated generic information technologies and optional cloud or edge computing platforms, which enable real-time, intelligent, and autonomous access, collection, analysis, communications, and exchange of process, product and/or service information, within the industrial environment, so as to optimise overall production value. This value may include; improving product or service delivery, boosting productivity, reducing labour costs, reducing energy consumption, and reducing the build-to-order cycle."

[12] describes the IIoT infrastructure defined in Figure 2.1 as a system consisting of:

- intelligent assets – i.e., applications, controllers, sensors and security components – that can sense, communicate and store information
- data communications infrastructure, e.g., the cloud;
- analytics and applications that generate business information from raw data; and people

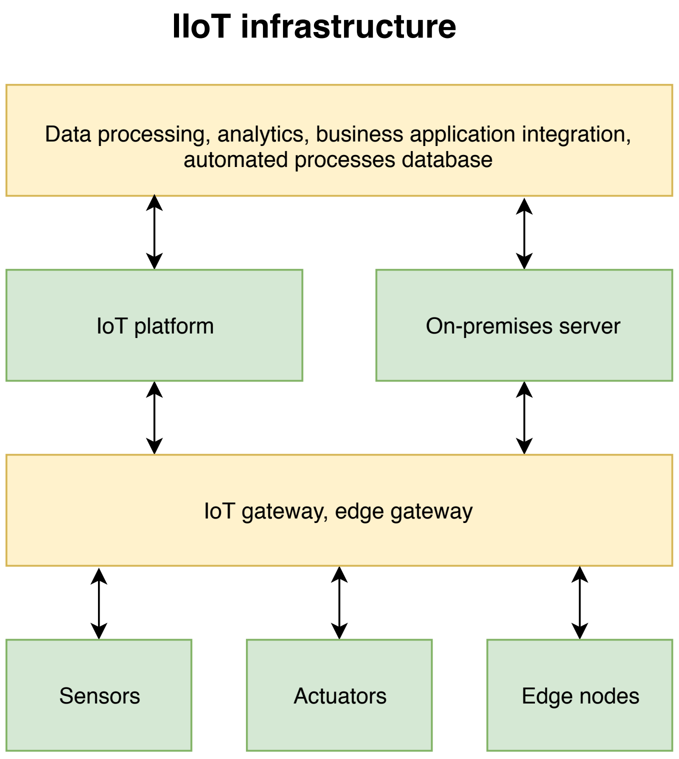

IIoT infrastructure
Data processing, analytics, business application integration, automated processes database
IoT platform
On-premises server
IoT gateway, edge gateway
Sensors
Actuators
Edge nodes

Figure 2.1: An overview of the IIoT infrastructure, from [techtarget-internetofthingsagenda-Industrial-Internet-of-Things-IIoT]

2.1.2 Applications of IIoT

According to [13], IIoT has the potential to transform traditional linear manufacturing supply chains into dynamic, interconnected systems that can more easily incorporate various parts of a industrial ecosystem. IIoT-technology can change the way that products are made and delivered, making factories more efficient, ensuring better safety for human operators, and, in some cases, saving millions of dollars. IIoT also provides great benefits in terms of improving operating efficiencies. If a machine malfuctions, connected sensors can automatically pinpoint where the issue is occurring and trigger a service request. IIoT can also help a manufacturer predict when a machine will likely break down or enter a dangerous operating condition before it ever happens. Thus among the flagship applications of IIoT we find highly critical operations such as monitoring of corrosion state of oil/gas pipelines or real-time monitoring of critical parts of a machinery, as well as less critical operations such as remote metering of water consumption. In addition, a number of non-industrial applications also benefit greatly from the technology behind IIoT, such as health state measurements of blood pressure/heartbeat of elderly; telemetry readings of status of oil/brakes/etc of cars on the move and occupancy measurements of parking in cities [14].

2.1.3 Enabling technologies

A key element to enabling IIoT is the advancements in wireless technology and ubiquitous low-cost sensor- and actuator-enabled devices which allow for low-power wireless networks with years of battery life, very high end-to-end reliability and support for determenistic latency - all of which are important requirements for industrial networks. These advances pave way for more flexibility and ease of deployment, as well as not having to rely on expensive cabling for networking industrial devices. The trend of industrial networks switching to IP technology, and the eventual convergence of Operational Technology (OT) and Information Technology (IT) through standards such as 6LoWPAN and 6TiSCH, also play a vital role in the realization of IIoT [15].

2.2 Wireless Sensor Networks

Wireless Sensor Networks are wireless communication networks made up of many sensor nodes deployed within range of each other forming a sensor field. This sensor field can then be used to gather data about and observe a small or large scale local phenomena e.g. for civilian, military, environmental, agricultural, industrial, or medical uses. WSNs possess self-organizing capabilities, which allows for very flexible deployments, as the position of the network nodes does not need to be engineered or predetermined. This makes it very useful for random deployment in inaccessible terrain or disaster relief operations, or in other scenarios where flexible, scalable, low-cost deployments are required. Because of its diverse applications, WSN is considered a popular field of research, and Akyildiz et. al [16] provides a comprehensive breakdown of challenges and research issues related to WSN.

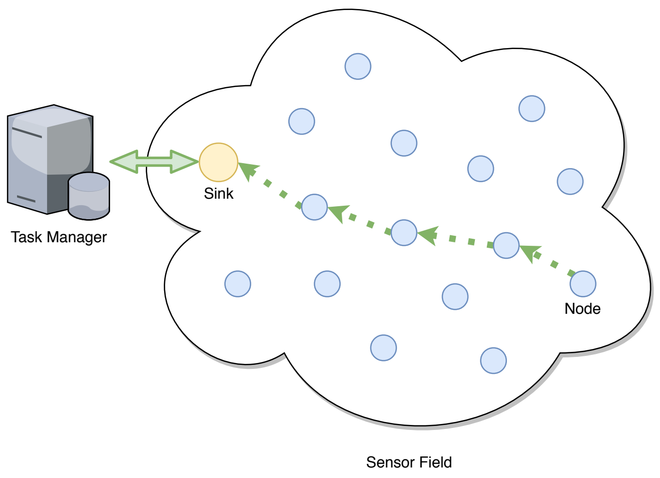


Figure 2.2: Illustration of a sensor field containing multiple sensor nodes and a sink node connected to a Task Manager. The green line represents a multi-hop path from a node towards the sink

A common WSN-model has the nodes periodically collect and transmit data to a sink node which serves as a data collection point, and which further communicates with an external Task Manager. Since not all sensor nodes are necessarily within communication range of the sink, nodes in a WSN can act as routers and forward data packets received from on-link neighbors towards the sink node over multiple hops.

WSN nodes are usually cheap, tiny, low-power embedded devices with sensing, data processing and communication capabilities. Most WSN nodes run on small rechargeable batteries, and thus to extend the target lifetime of a node (which can be in the range of several years) these devices end up having very limited memory and computational capacities. Efficient use of the limited memory available in sensors is required; this means heavily constraining or even opting-out of memory consuming features like routing tables or security.Hardware constraints, fault tolerance, scalability and power consumption are important factors that have driven the design of WSNs. Production cost is also an important design factor. Sensor networks can be made up of hundreds of nodes, and the price per sensor node must be kept low for WSNs to be cost-justified compared to other solutions.

Because of the limited amount of power available per node, the protocol stack used by the nodes is required to enable efficient power consumption. As the radio usually consumes the most energy in a WSN node, the largest gain in energy efficiency can be achieved at the MAC and PHY layer. By ensuring that nodes use communication resources sparingly and efficiently, as well as avoiding retransmissions by preventing collisions in the wireless medium, energy consumption can be kept to a minimum.

The limited power supply and the need for a resource- and power-efficient MAC-layer means that well known wireless standards such as IEEE 802.11 (aka WiFi), while capable of high data throughput and range, consume too much energy to be a viable alternative for WSNs. Instead, WSN nodes employ wireless standards specifically designed for low-power wireless communication, such as the IEEE 802.15.4 standard for Low-Rate Wireless Networks. This standard provides low-cost, short-range, low-power and low data-rate communication for sensor networks, and is utilized in WSN standards/stacks such as 6LoWPAN, ZigBee, WirelessHART and ISA100.11a. Table 2.1 provides some more detail on these mentioned standards and their differences.

Most WSN technologies operate in the ISM band (Industrial, Scientific and Medical radio band), which are internationally reserved for industrial, scientific and medical purposes. This means that WSN usually has to share the medium with other wireless devices operating on the ISM band which then become sources of interference and can cause communication problems.

Standard	**Topology**	**Battery life (days)**	**Network nodes**	**Max Throughput**	**Range (m)**
ZigBee	Mesh	100-1000+	255	250 Kbps	10-100
6LoWPAN	Mesh	100-365+	65536	250 Kbps	1-100
WirelessHART	Mesh	760+	200	250 Kbps	1-100
ISA100.11a	Mesh, Star	1000+		250 Kbps	100

Table 2.1: Key differences of popular 802.15.4 based WSN standards, adapted from [17]

2.2.1 Industrial Wireless Sensor Networks

An Industrial Wireless Sensor Network (IWSN) is a type of WSN that is employed in an industrial setting in order to monitor industrial equipment. Traditionally, industrial communication has relied on using wired networks. These networks, while reliable, require the installation of expensive- regularly-maintained cables and are thus not widely implemented in industrial plants. However, with the recent technological advances in WSN, the realization of low-cost embedded wireless industrial systems have become feasible [18].

A IWSN system works similarly to a traditional non-industrial WSN; Wireless tiny sensor nodes are installed on industrial equipment that monitor parameters critical to equipment efficiency. This can be a combination of measurements such as vibration, temperature, pressure, and power quality. This information is then wirelessly transmitted to a sink node that analyzes the data from each sensor. Any potential problems are notified to the plant personnel as an advanced warning system. This enables plant personnel to repair or replace equipment, before their effciency drops or they fail entirely. In this way, catastrophic equipment failures and the associated repair and replacement costs can be prevented.

The collaborative nature of IWSNs brings several advantages over traditional wired industrial monitoring and control systems, including self-organization, rapid deployment, flexibility, and inherent intelligent processing capability. In this regard, IWSN plays a vital role in creating a highly reliable and self-healing industrial system that rapidly responds to real-time events with appropriate actions. The existing and potential applications of IWSNs span a very wide range, including building automation, industrial process automation, electric-utility automation, automatic meter reading, and inventory management [18].

Challenges	**Design Goals**
Resource constraints	Resource efficient design
Dynamic topologies and harsh environmental conditions	Adaptive network operation
Quality of service requirements	Application-specific design and time synchronization
Data redundancy	Data fusion and localized processing
Packet errors and variable link capacity	Fault tolerance and reliability
Security	Secure design
Large scale deployment and ad hoc architecture	Low cost and small sensor nodes, and self configuration and organization
Integration with Internet and other wireless technologies	Scalable architectures and efficient protocols

Table 2.2: Challenges versus design goals in IWSNs, adapted from [17]

The industrial domain introduces a set of challenges to WSN, detailed in Table 2.2. In particular, industrial networks have more stringent requirements when it comes to latency and reliability. While traditional wireless sensor network applications have been assumed as delay-tolerant, real-time requirements are of utmost importance in industrial (wireless) networks, since delayed sensor data could be considered useless or even detrimental to the system being monitored. The industrial environment can inhibit characteristics that make these requirements hard to meet, such as signal attenuation from dust, heat and water, as well as multipath fading and electromagnetic interference from other wireless devices. These characteristics impact the reliability of the wireless links, which in turn negatively affects the performance of the IWSN. To combat these characteristics, IWSN standards usually employ TDMA and FDMA techniques. TDMA assures that nodes follow a precice schedule that dictates whether they should send, receive or idle by synchronizing communication over atomic units of time, and FDMA seeks to increase the robustness of the wireless links by introducing frequency diversity, allowing multichannel communication and channel hopping which helps mitigate the effects of multipath fading and radio interference. WirelessHART [2] and ISA100.11a [3] are two popular IWSN standards which use TDMA together with channel hopping to make them robust to interference in harsh industrial environments [19]. While both are based on the same basic WSN technology of IEEE 802.15.4, they have fundamental differences in design philosophy. WirelessHART opts for ease of deployment and multivendor compatibility, while ISA100.11a chooses flexibility and scalability as its key design features [20].

2.3 The IEEE 802.15.4 standard for Low-Rate Wireless Networks

IEEE 802.15.4 is a standard for low-rate, low-power, and low-cost Personal Area Networks (PANs)[a-step-towards]. The standard defines the physical (PHY) and medium access control (MAC) layer specifications for low-data-rate wireless connectivity with fixed, portable and moving devices with very limited energy consumption requirements. The standard provides for ultra low complexity, ultra low cost, ultra low power consumption, and low data rate wireless connectivity among inexpensive devices [4].

2.3.1 Architecture

The IEEE 802.15.4 architecture is defined in terms of a number of blocks in order to simplify the standard. These blocks are called layers. Each layer is responsible for one part of the standard and offers services to the higher layers. The interfaces between the layers serve to define the logical links that are described in this standard.

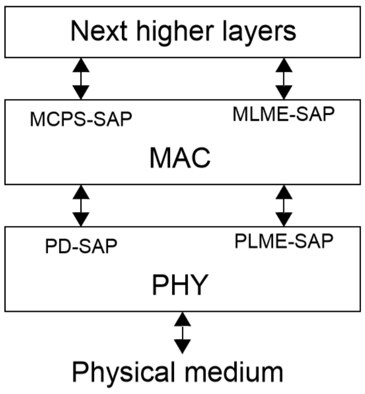


Figure 2.3: LR-WPAN device architecture, from [4]

An LR-WPAN device comprises at least one PHY, which contains the radio frequency (RF) transceiver along with its low-level control mechanism, and a MAC sublayer that provides access to the physical channel for all types of transfer. Figure 2.3 shows these blocks in a graphical representation.

2.3.2 Topologies

An IEEE802.15.4 LR-WPAN can operate in either of two topologies: the star topology or the peer-to-peer topology. Both are shown in Figure 2.4. In the star topology, the communication is established between devices and a single central controller, called the PAN coordinator. A device typically has some associated application and is either the initiation point or the termination point for network communications. The PAN coordinator is the primary controller of the PAN. The PAN coordinator will often be mains powered, while the devices will likely be battery powered. Applications that benefit from a star topology include home automation, personal computer (PC) peripherals, games, and personal health care.

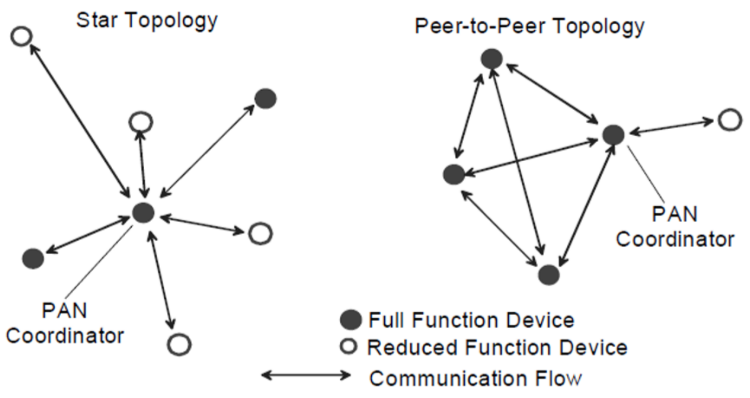


Figure 2.4: IEEE 802.15.4 std: supported topologies, from [4]

The peer-to-peer topology also has a PAN coordinator; however, it differs from the star topology in that any device is able to communicate with any other device as long as they are in range of one another. Peer-to-peer topology allows more complex network formations to be implemented, such as mesh networking topology. Applications such as industrial control and monitoring, wireless sensor networks, asset and inventory tracking,intelligent agriculture, and security would benefit from such a network topology. A peer-to-peer network allows multiple hops to route messages from any device to any other device on the network.

2.3.3 Device types

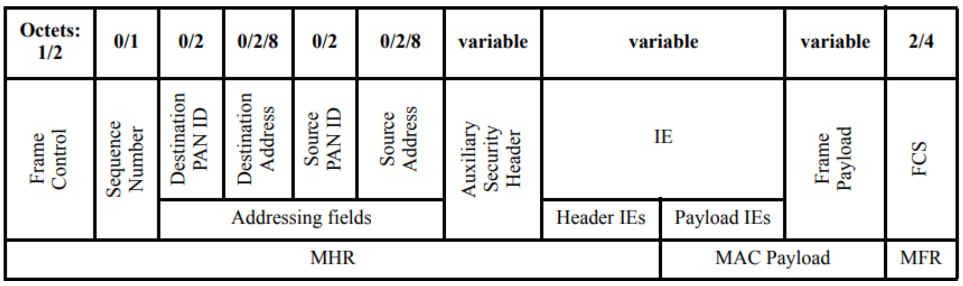
Two different device types can participate in an IEEE802.15.4 network: a full-function device (FFD) and a reduced-function device (RFD). An FFD is a device that is capable of serving as a personal area network (PAN) coordinator or a coordinator. An RFD is a device that is not capable of serving as either a PAN coordinator or a coordinator. An RFD is intended for applications that are extremely simple, such as a light switch or a passive infrared sensor; it does not have the need to send large amounts of data and only associates with a single FFD at a time. Consequently, the RFD can be implemented using minimal resources and memory capacity [4].

2.3.4 MAC

The MAC sublayer provides an interface between the next higher layer and the PHY, and enables the transmission and reception of MAC protocol data units (MPDUs) across the PHY data service. The features of the MAC sublayer are beacon management, channel access, GTS management, frame validation, acknowledged frame delivery, association, and disassociation. In addition, the MAC sublayer provides hooks for implementing application-appropriate security mechanisms.

The legacy IEEE 802.15.4-2006 standard offers the use of either slotted and unslotted CSMA-CA as a MAC protocol. Following IEEE 802.15.4e, detailed later in this section, the MAC sublayer allows the operation of different MAC behaviors to support specific application domains. This master thesis will only consider the Time Slotted Channel Hopping (TSCH) behavior of the 802.15.4e MAC, and as such will not detail any of the other modes of operation.

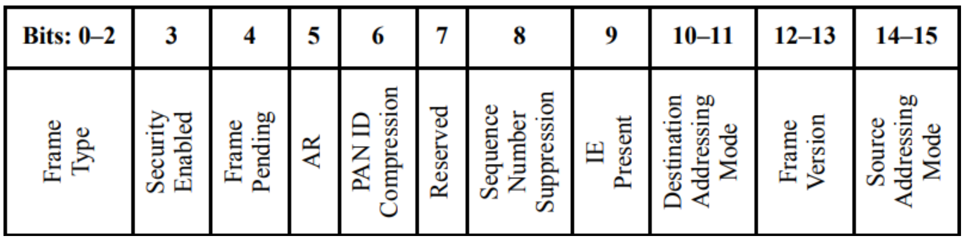
General MAC frame format



Octets: 1/2	0/1	0/2	0/2/8	0/2	0/2/8	variable	variable		variable	2/4
Frame Control	Sequence Number	Destination PAN ID	Destination Address	Source PAN ID	Source Address	Auxiliary Security Header	IE		Frame Payload	FCS
		Addressing fields					Header IEs	Payload IEs		
MHR								MAC Payload		MFR

Figure 2.5: Format of the MAC frame, from [4]

Frame Control field



Bits: 0–2	3	4	5	6	7	8	9	10–11	12–13	14–15
Frame Type	Security Enabled	Frame Pending	AR	PAN ID Compression	Reserved	Sequence Number Suppression	IE Present	Destination Addressing Mode	Frame Version	Source Addressing Mode

Figure 2.6: Format of the Frane Control field, from [4]

Frame pending field

The frame control field includes the Frame Pending field, which can be used to indicate to a receiving device that another packet is pending transmission to this destination. The frame pending field is also referred to as the "pending bit". The frame control field is used for signaling in the idle listening reduction mechanism proposed for this thesis, which is introduced in chapter 3.

2.3.5 PHY

The PHY layer provides an interface between the MAC sublayer and the physical radio channel, via the RF firmware and the RF hardware. The PHY is responsible for a number of tasks such as activation and deactivation of the radio tranceiver, Clear Channel Assessment (CCA) for the CSMA-CA protocol, channel frequency selection, and data transmission and reception in the form of PHY protocol data units (PPDUs).

2.3.6 IEEE 802.15.4e

While the legacy IEEE 802.15.4-2006 defined a MAC and PHY layer tailored specifically for use in LLNs, several studies highlighted flaws of the standard that made it unsuitable for use in critical scenarios, such as those found in industrial applications. These flaws are characterized by [21] as:

- Unbounded delay: the MAC protocol of IEEE 802.15.4-2006 cannot provide a bound on maximum delay for data to reach its destination. This makes it unsuitable for applications that have requirements of timeliness and deterministic latency.

- Limited communication reliability: the slotted CSMA-CA algorithm used in IEEE 802.15.4-2006 Beacon Enabled (BE) mode provides a very low delivery ratio, even with a small node count. Similar issues can also be found in the Non-Beacon Enabled (NBE) mode.

- No protection against interferences/fading: because the IEEE 802.15.4-2006 MAC only uses a single channel and lacks a frequency hopping mechanism, it is subject to common phenomena in wireless networks such as interference and multi-path fading, which can cause frequent instabilities in the network.

- Powered relay nodes: to overcome limitations in the standard, intermediate relay nodes in IEEE 802.15.4-2006 multi-hop networks keeps their radio on at all times, causing large energy consumption.

To address these issues and better meet the demands of the industrial market, the IEEE formed the 802.15.4e Task Group, which aimed at defining an improved low-power MAC layer that could meet the stringent requirements of industrial applications. Building on concepts found in industrial WSN standards WirelessHART and ISA100.11a, the resulting IEEE 802.15.4e-amendment [22] describes several improvements to the MAC layer of the 802.15.4-2006 standard that makes it suitable for use in industrial applications. These improvements include the implementation of features such as slotted access with shared and dedicated slots and multi-channel communication with frequency hopping, together with several general functional improvements. After its approval in 2012, the 802.15.4e-amendment was later included into IEEE 802.15.4-2015.

The 4e amendment defines a new MAC scheme called Time Slotted Channel Hopping (TSCH), which aims to deliver deterministic latency and multichannel operation to applications that require it, such as the ones found in the industrial domain. Addressing the need for reliability, timeliness and multi-hop communication within industrial networks, TSCH is considered a key enabler for IIoT/IWSNs and an important addition to the 802.15.4 MAC.

2.4 Time Slotted Channel Hopping

Time Slotted Channel Hopping (TSCH) is one of the new MAC behaviour modes defined by the 802.15.4e standard. It is a medium access technique designed to be used in low-power deterministic networking applications, such as in process automation for oil and gas, pharmaceuticals, green energy and more. It is considered by [1] as the de-facto technology for highly reliable, low-power wireless sensor networking.

2.4.1 Background

TSCH takes inspiration from industrial networking standards WirelessHART and ISA100.11a by combining time slotted access with multi-channel and channel hopping capabilities. The time slotted aspect of TSCH allows for a network with a higher potential throughput due to the elimination of collisions happening between transmitting nodes. This approach also provides deterministic latency to applications that require it, as well as a low duty cycle for increased energy efficiency. Multi-channel and channel hopping capabilities allows nodes to communicate in the same time slot without risk of collisions, and mitigates the effects of interference and multipath fading [A Step Towards the Internet of Things].

2.4.2 Topology

TSCH is topology independent, and can be used in star, tree, partial- or full-mesh configurations. TSCH is particularly well-suited for multi-hop networks because of its multi-channel operation, where frequency hopping allows for efficient use of the available channel resources.

2.4.3 Synchronized communication

Nodes in a TSCH network synchronize their communication in a TDMA-manner using *slotframes* divided into short intervals of time called *timeslots*. Timeslots are fixed units of time which represent a communication opportunity for a node within the slotframe.

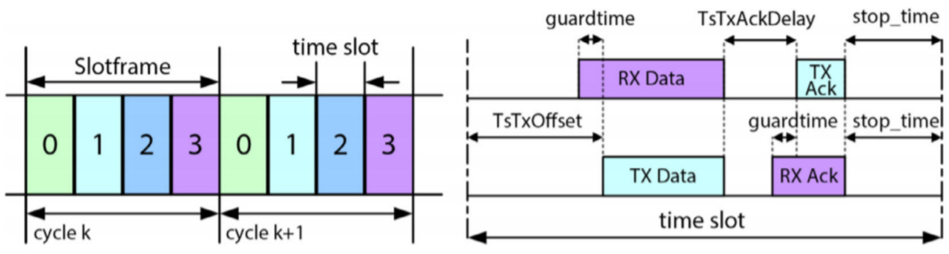


Figure 2.7: TSCH Slotframe (left) and Timeslot (right), from [21]

Slotframe

The slotframe is divided into timeslots, and repeats over time. The number of timeslots in a slotframe is dictated by the slotframe size, which can range from 10s to 1000s depending on the application. For each timeslot, the slotframe dictates whether the node should sleep or transmit/receive. In the case of the latter, it also dictates a set *channelOffset* used for frequency selection.

Cells

A combination of a *slotOffset* and a *channelOffset* defines a *cell* within a slotframe. The slotOffset refers to the position of a timeslot within a slotframe, and the channelOffset indicates which frequency to communicate on.

Absolute Slot Number

The Absolute Slot Number (ASN) is a global value that is defined by the number of timeslots that has elapsed since the start of the network or an arbitrary start time determined by the PAN coordinator. The ASN is known by all nodes in the network and dictates the current timeslot all nodes are operating in. The ASN is globally incremented after the passing of a timeslot.

Timeslot structure

The structure of a timeslot is depicted in Figure 2.7 (right). Each timeslot allows a node to send a maximum-size data frame and receive the related acknowledgement. Data packets are transmitted after TsTxOffset µs, and listening starts GuardTime µs before the end of TsTxOffset. This allows for slight desynchronization. In addition, if the reception of the packet does not begin within GuardTime µs after TsTxOffset, the node disables its radio to save energy.

The duration of a time slot is not defined by the standard, but 10 ms is often used, as it gives time for transmission of a maximum-length frame of 127 bytes and the corresponding ACK, and still leaves time for radio turnaround, packet processing and security operations (assuming 250 kbps 802.15.4).

2.4.4 Links

A link is defined as the pairwise assignment of a directed communication between nodes in a given timeslot on a given channelOffset. TSCH supports two different link types: dedicated links, and shared links. A dedicated link is a contention-free link reserved only for a single node pair to use, while a shared link contains multiple node pairs that compete for channel using CSMA-style back-off.

Retransmission

For a dedicated link, if an acknowledgement is not received within a predefined timeout within the timeslot, the retransmission of the data frame is deferred to the next time slot assigned to the same destination. In the case of transmission failure over a shared link, TSCH uses a CSMA-style retransmission algorithm with a backoff defining a random amount of shared cells to wait before a retransmission attempt.

2.4.5 Channel hopping

For any scheduled slotframe cell, the frequency f used to communicate is decided by Eq 2.1

$$f = F[(ASN + channelOffset)\%N_{channels}] \tag{2.1}$$

where $N_{channels}$ is the number of available channels, function F a lookup table of available channels, ASN the current network ASN and $channelOffset$ the channeloffset of the active cell. This functionality ensures successive packets between links are transmitted using different frequencies, which can give a higher probability of success in case of retransmissions due to fading or interference. There are intially 16 channels available for communication, each identified by an integer value in the range of [0, 15]. Channels can be blacklisted in the case of poor performance.

2.4.6 Scheduling

TSCH relies on a link schedule that defines when and how nodes communicate with each other. This link schedule can be illustrated as a matrix with width equal to the slotframe size, and a height equal to the number of available frequencies (channels). The allocated cells of the link schedule matrix represent a link between a pair of nodes in a given timeslot and channel offset. Figure 2.8 illustrates a TSCH-schedule over a example topology.

Maintaining and deriving the link schedule is not part of the IEEE 802.15.4 standard scope, and so this exercise is left to the upper layers. Creating an optimal link schedule for a TSCH network is not always a trivial task, especially in large networks with multi-hop topology and in dynamic networks where the topology changes over time. Scheduling in TSCH networks is an active field of research, and many proposals for scheduling algorithms have appeared in literature [23].

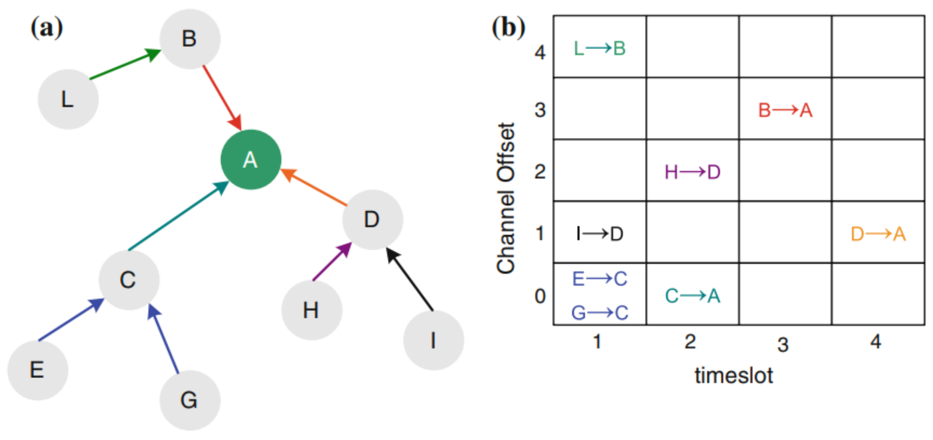


Figure 2.8: A sensor network with a tree-topology (a) with a possible link schedule for data-collection (b), from [23]

2.5 6TiSCH - IPv6 over the TSCH mode of IEEE 802.15.4e

6TiSCH [6] is a protocol stack created by the IETF 6TiSCH Working Group that aims to enable IPv6 for TSCH LLNs. Driven by the trend of industrial networks to utilize IP-technology, its purpose is to allow IT and OT convergence while meeting the requirements of low-power wireless deterministic applications.

2.5.1 Protocol stack

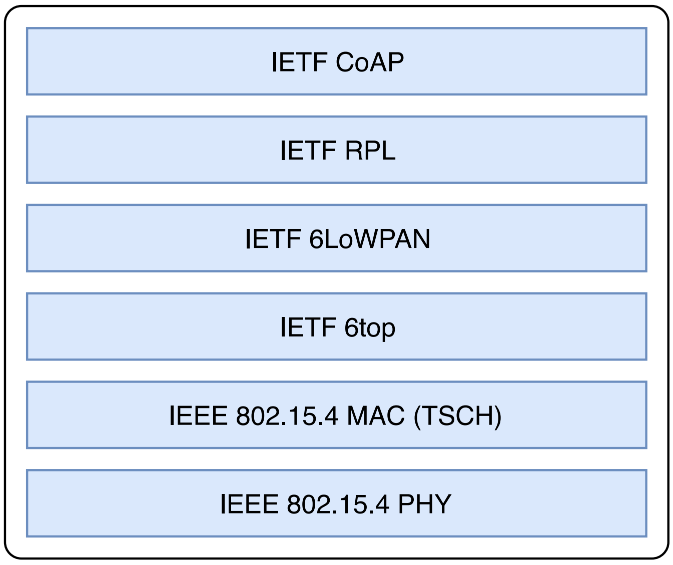


Figure 2.9: The 6TiSCH protocol stack

The 6TiSCH protocol stack is composed of standards developed by both the IEEE and the IETF. From top to bottom, these are the Constrained Application Protocol (CoAP), Routing Protocol for Low-Power and Lossy Networks (RPL), IPv6 Header Compression (6LoWPAN), 6TiSCH 6top, and the IEEE 802.15.4 MAC & PHY operating in the TSCH mode.

2.5.2 6TiSCH operation sublayer (6top)

6TiSCH defines the 6top sublayer as a logical link control just above the IEEE 802.15.4 MAC layer. 6top provides the link abstraction that is required for IP operations, and offers a management API that enables an external management entity to schedule TSCH cells and slotframes.

6top protocol (6P)

The 6top protocol (6P) specifies a protocol for an external entity to manage the TSCH schedule of a node. This external entity can either be a neighbor node or some kind of a management entity, allowing for both distributed and centralized schedule management.

Schefuling Function

A Scheduling Function (SF) is an entity on the mote that is responsible for maintaining a distributed TSCH schedule. Many different proposals for SFs exist in literature regarding 6TiSCH. The *6TiSCH Minimal Scheduling Function* (MSF) is an example of a SF proposed by the 6TiSCH WG that is designed to operate in a wide range of application domains.

2.5.3 Scheduling in 6TiSCH

As 6TiSCH uses TSCH to provide deterministic networking capabilities, it is dependent on mechanisms to build and maintain the TSCH-schedules of the nodes in the 6TiSCH LLN. 6TiSCH identifies four ways a schedule can be managed: static scheduling, neighbor-to-neighbor scheduling, Centralized (or Remote) scheduling, and hop-by-hop scheduling.

Static Scheduling

A static schedule is configured for the whole network and is distributed through the native methods in the TSCH MAC layer. Other scheduling operations can co-exists with the static schedule on the same 6TiSCH network. The Miminal 6TiSCH Configuration is an example of a static schedule which is used for network bootstrapping. This schedule is pre-established, for instance decided by a network administrator based on operational needs.

Neighbor-to-Neighbor Scheduling

Refers to dynamic bandwidth adaptaion of the links that are used for traffic between neighbor nodes. A node uses Scheduling Functions (SFs) to add, update, and remove cells in its own and its peer's schedules through the 6P protocol.

Centralized scheduling

Refers to central computation of a schedule by a Network Management Entity (NME), which may work together with a Path Computation Entity (PCE). Enables Traffic Engineering with deterministic properties.

Hop-by-hop scheduling

Refers to the possibility to reserve cells along a path for a particular flow using a distributed scheduling mechanism.

2.5.4 Tracks

The 6TiSCH architecture introduces the concept of a Track, which is a directed and deterministic path from a source node to a destination node within a 6TiSCH LLN. Tracks are made up of bundles of tx- and rx-cells installed at each mote along the multi-hop path. Constrained resources such as memory buffers are reserved for that track in intermediate 6TiSCH nodes to avoid loss related to limited capacity. A 6TiSCH node along a track not only knows which bundles of cells it should use to receive packets from a previous hop, but also knows which bundle(s) it should use to send packets to its next hop along the Track. Tracks can be reserved either by a remote PCE or by using a distributed mechanism such as Hop-by-hop scheduling. Figure 2.10 illustrates a simple track. [6] lists some specific benefits of using tracks to forward a packet from a source to destination:

1. Track forwarding is a Layer-2 forwarding scheme, which introduces less process delay and overhead than a Layer-3 scheme. Thus LLN devices can save more energy and resources, which is important for such devices.

2. Because bundles of cells have alredy been reserved for communications between nodes on each hop of the track, throughput and maximum latency is guaranteed and the jitter is maintained small.

3. By knowing the scheduled timeslots of incoming and outgoing bundle(s), nodes on a track can save energy by staying asleep during inactive slots.

4. Tracks are protected from interfering with one another if a cell belongs to at most one Track, and congestion loss is avoided if at most one packet can be presented to the MAC to use that cell. Tracks enhance the reliability of transmissions and thus further improve the energy consumption in LLN Devices by reducing the chances of retransmission.

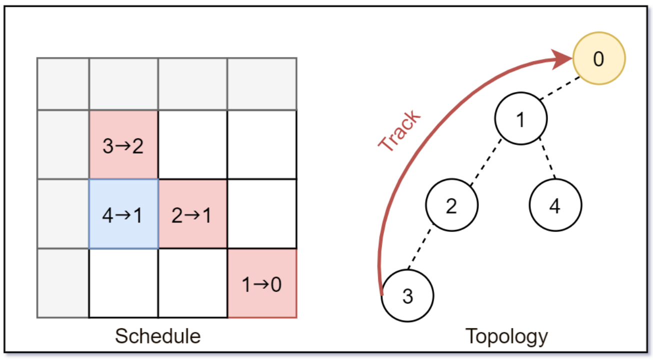


Figure 2.10: A simple illustration of a 6TiSCH track. The scheduled cells marked red are part of a track from node 3 to the sink

Overprovisioning

Multiple cells may be scheduled in a track for the transmission of a single packet, in order to allow multiple transmission opportunities per packet(s). The number of cells in a bundle per hop along a track introduces a tradeoff between energy usage and bandwidth. If the size of the bundles is configured to fit an average amount of bandwidth, peak traffic is dropped. If the size is configured to allow for peak emissions, energy is be wasted idle listening. A track can be considered optimal when the size of the track bundles per hop are such that both the energy wasted in idle listening and the packet drops due to congestion loss are minimized, while packets are forwarded within an acceptable latency.

Track types

The 6TiSCH architecture draft defines two track types: serial tracks and complex tracks. In a serial track a bundle of cells that are set to receive is uniquely paired to a bundle of cells set to transmit. Together these bundles form a Layer-2 forwarding method which can be used regardless of the network layer protocol. A serial track is thus formed end-to-end as a succession of paired bundles. The bundles may be computed so as to accommodate both variable rates and retransmissions through overprovisioning [6].

A complex track, as opposed to a serial track, is shaped as a directed acyclic graph towards one or more destinations to support multi-path forwarding and route around failures. Thus, complex tracks can employ Deterministic Networking techniques such as Packet Replication, Elimination and Ordering Functions (PREOF). The 6TiSCH architecture also states that a complex track can be part of a larger DetNet End-to-end path that can extend beyond the 6TiSCH LLN.

Track Forwarding

Forwarding along a track can be seen as a Generalized Multi-protocol Label Switching (G-MPLS) operation in that information used to switch a frame is related to properties of the way the packet was received, eg. which particular cell that received the packet. As a result, as long as the TSCH MAC accepts a frame, that frame can be switched regardless of the protocol, whether this is an IPv6 packet, a 6LoWPAN fragment, or a frame from an alternate protocol such as WirelessHART or ISA100.11a.

Cell Reuse

A track TX-cell that is not needed for the current iteration of a slotframe may be reused for other packets. When all the frames that were received for a given track have been transmitted, any available track cells can be reused for upper layer traffic for which the next-hop router matches the next hop along the track. In addition, if there are not enough TX-cells to accomodate track traffic, frames can be placed in the bundle that is used for Layer-3 traffic towards the next hop along the track.

2.6 Analytical

This section will present formulas for the calculation of some aspects related to the TSCH and other aspects of the thesis.

2.6.1 Packet latency

Since time in TSCH is divided into timeslots lasting a specified amount of time, packet latency in TSCH can be calculated.

As such, the maximum latency of a packet traveling from a source to a destination can be given by

$$L_{max} = S \cdot N \cdot R \cdot T_{slot} \tag{2.2}$$

where S is the slotframe size, N is the number of hops to the destination, R is the maximum amount of retransmissions per slot and T_{slot} is the length of a timeslot in seconds.

Similarily, we can define the minimum latency for a packet by

$$L_{min} = N \cdot T_{slot} \tag{2.3}$$

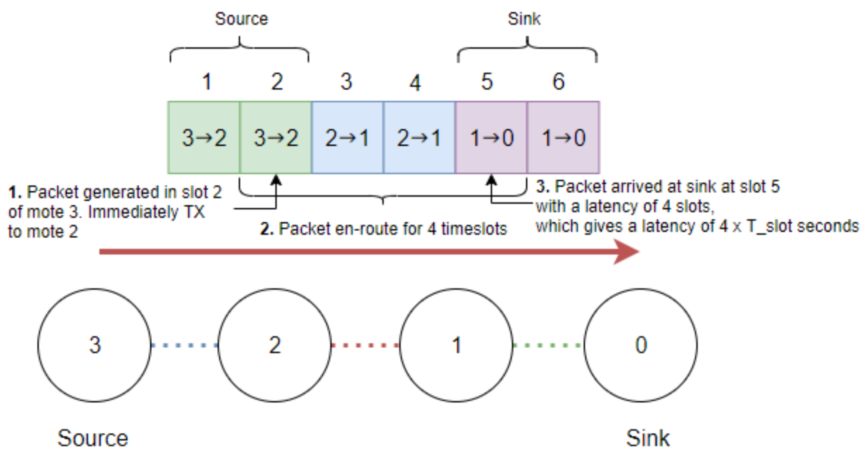
Latency with overprovisioned cells

Calculating latency becomes more complicated for a network with a specific amount of overprovisioned cells per hop, such as in the scenarios defined in the thesis. However, the minimum latency in such a topology can be defined as

$$L_{min} = 2T_{slot} + \sum_{n=1}^{N-2} n \cdot C \cdot T_{slot}, N \geq 2 \tag{2.4}$$

where C is the amount of overprovisioned cells per hop.

For a single hop, the minimum latency is defined by the value of T_{slot} alone. Figure 2.11 illustrates an example of minimum packet latency in a simple TSCH network with overprovisioned cells.



Source
Sink
1
2
3
4
5
6
3→2
3→2
2→1
2→1
1→0
1→0
1. Packet generated in slot 2 of mote 3. Immediately TX to mote 2
2. Packet en-route for 4 timeslots
3. Packet arrived at sink at slot 5 with a latency of 4 slots, which gives a latency of 4 x T_slot seconds
3
2
1
0
Source
Sink

Figure 2.11: Illustration of the minimium packet latency occuring in an example TSCH network featuring a single overprovisioned cell per hop with 3 hops between source and destination.

2.6.2 Packet generation

Random packet generation in the thesis is done following a poisson distribution, where λ is defined as the rate of packets generated per timeslot.

The probability of generating x number of packets in a single slotframe is defined as

$$P(x) = \frac{e^{-\lambda}\lambda^{x}}{x!} \cdot S \tag{2.5}$$

where S is the length of a slotframe.

The expected total amount of packet generated in a simulation run is defined as

$$P_{total} = \lambda \cdot S \cdot N_S \tag{2.6}$$

where S is the length of a slotframe and N_S is the amount of slotframes where the mote is generating packets.

Periodic packet generation

If given a period in number of slotframes per a packet generation event, the length in slotframes of which to generate packets and an amount of packets generated per period, the total amount of packets generated can be calculated using

$$P_{total} = \frac{N_P \cdot N_S}{T} \tag{2.7}$$

where N_S is the amount of slotframes where the mote is generating packets, N_P is the amount of packets generated per period, and T is the period per packet generation event in number of slotframes.

Chapter 3

Proposal

This chapter will detail Track Resource Adaptation (TRA), which is this thesis' proposed method for energy conservation through reduced idle listening in tracks with overprovisioned cells.

3.1 Track Resource Adaptation

Track Resource Adaptation defines a mechanism for reducing idle listening occuring on overprovisioned cells in 6TiSCH tracks. The idea behind the mechanism was based partly upon [8] in how it considers a way to enable and disable overprovisioned cells based on signaling between neighbor nodes. TRA is envisioned in this thesis as an entity residing on each mote along a track from source to destination.

3.1.1 Mechanism

TRA will disable or enable overprovisioned cells belonging to a track in a node's TSCH schedule based on the presence of the Frame Pending field (pending bit) of the received frames. Figure 3.1 illustrates the basic mechanism behind TRA. A node which utilizes TRA will read the pending bit of a received frame to decide whether the following track RX-cells should be enabled for frame reception. For a frame arriving on an enabled track RX-cell, if the pending bit is set (=1), the successive track RX-cells in the schedule will be enabled for the current slotframe iteration. If the pending bit is not set (=0), the following track RX-cells are disabled, and a node will not perform listening during the scheduled timeslots. In this way, TRA provides a possibility for nodes hosting a track bundle to reduce idle listening when there are no expected incoming frames for this track. The state of Enabled/Disabled cells is only kept for the current slotframe iteration, and will reset on the start of the next.

Disabling/enabling cells

In TRA, the term "Disabling" or "Enabling" a cell refers to whether a allocated cell is available for TSCH to use for RX/TX. E.g. a disabled cell

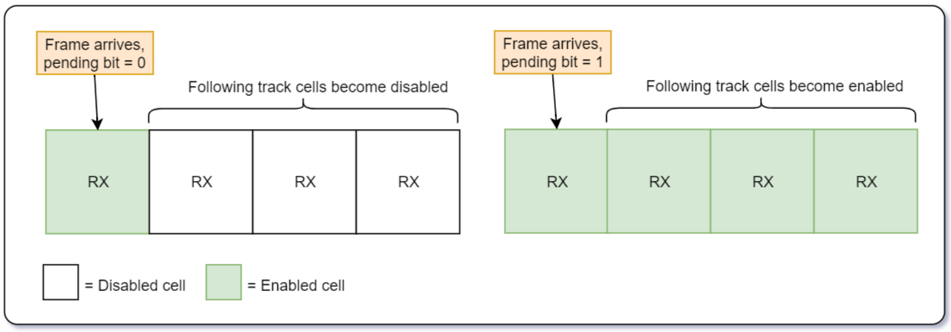

Frame arrives, pending bit = 0
Following track cells become disabled
RX
RX
RX
RX
Frame arrives, pending bit = 1
Following track cells become enabled
RX
RX
RX
RX
= Disabled cell
= Enabled cell

Figure 3.1: TRA mechanism example for a node with 3 overprovisioned track cells.

will not consume radio energy during its scheduled timeslot. Enabling or disabling a cell does not have any effect on its allocation – a disabled cell will still be considered allocated in the TSCH schedule.

Disabling TX-cells

Since TRA will disable overprovisioned RX-cells on the receiver side of a link, it is also necessary to disable the related TX-cells on the sender side in order to prevent the node from transmitting when there is no receiver cell listening. This mechanism is implemented in a manner similar to the receiver side, where the successive TX-cells are enabled/disabled based on the presence of the pending bit of a successfully transmitted frame.

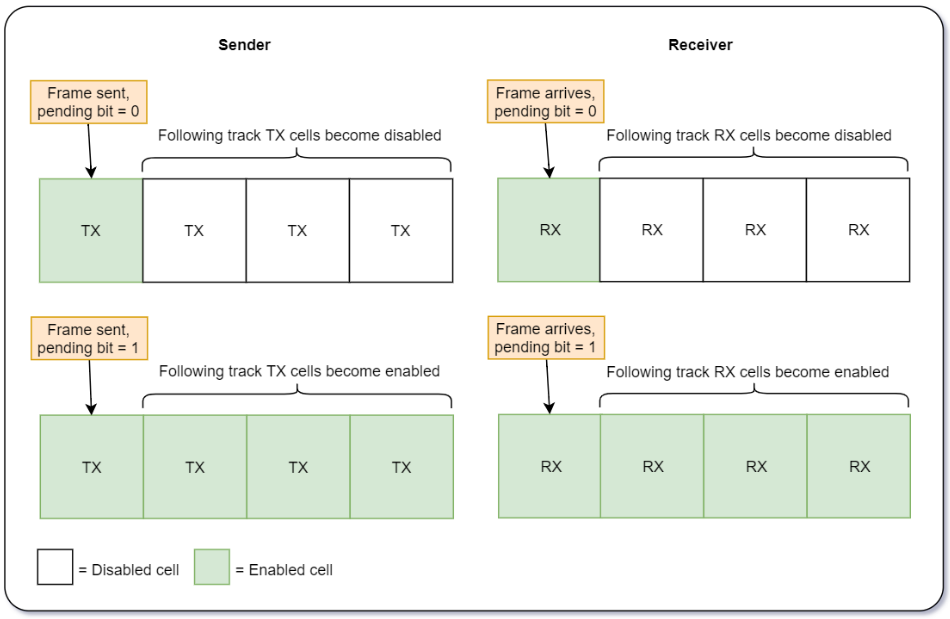

Sender
Receiver
Frame sent, pending bit = 0
Following track TX cells become disabled
TX
TX
TX
TX
Frame arrives, pending bit = 0
Following track RX cells become disabled
RX
RX
RX
RX
Frame sent, pending bit = 1
Following track TX cells become enabled
TX
TX
TX
TX
Frame arrives, pending bit = 1
Following track RX cells become enabled
RX
RX
RX
RX
= Disabled cell
= Enabled cell

Figure 3.2: TRA mechanism example for the sender and receiver side of a hop along a track.

3.1.2 Variants

The thesis defines two variants of TRA: *All Listen* and *One-Shot*. Both share the same functionality, but differ in the amount of cells initially enabled at the start of a slotframe iteration. In the All Listen variant, all track cells are enabled at the beginning of the slotframe. Cells are only disabled when a frame that contains an unset pending bit is received. This allows for retransmission opportunities in cases of low link quality, but may come at the cost of more cells spent idle listening.

For the One-shot variant of TRA, only a single cell is enabled at the beginning of a slotframe. This means that a frame will only have one transmission opportunity to enable the rest of the cells based on the presence of the pending bit. This can offer a higher amount of idle listening reduction, but might adversely impact the latency and reliability of the network, as there are no retransmission opportunities in case the packet fails in the first cell.

3.1.3 Usage of the pending bit

The frame pending field, also called the pending bit, is part of the frame control field of the 802.15.4 MAC frame. According to [ieee802.15.4], when operating in TSCH mode the frame pending bit can be set to one to indicate that the recipient should await a new transmission on the next timeslot and on the same channel if there is no link scheduled. At all other times, it shall be set to zero on transmission and ignored on reception. TRA disables the functionality of scheduling extra opportunistic transmissions for frames belonging to a track, and instead uses the pending bit for enabling or disabling already allocated track cells.

3.1.4 Use cases

TRA could be useful in a number of scenarios relating to overprovisioning. For example, in cases where a node has contiguous overprovisioned cells to a neighbor node (e.g. to meet Estimated Number of Transmissions, also known as ETX), the All Listen variant of TRA can be used to prevent idle listening in cases where the transmission is successful in the first cell. On the other hand, the One Shot variant of TRA could offer a high level of idle listening reduction in cases where overprovisioning primarily stems from trying to allocate bandwidth for traffic spikes in variable-rate traffic applications. The exact benefit of TRA (in the form of the two variants) is up for evaluation in this thesis, where reduction in idle listening is compared to eventual drawbacks introduced. This means examining how TRA behaves in different scenarios, with specific traffic flows and wireless link properties, and identifying the situations where TRA might offer a benefit in reducing idle listening, and thus reduce energy consumption.

Chapter 4

Method

This chapter will describe the evaluation method used in the project. Some thoughts surrounding evaluation of a WSN is presented, along with a description of of Discrete Event Simulation and the process behind selecting a simulation tool. Lastly, an introduction to the 6TiSCH Simulator is given.

4.1 Evaluation of Wireless Sensor Networks

Analysing Wireless Sensor Network designs can be a difficult task. WSNs can often be too complex to model analytically because of the impact the deployment environment has on the generated traffic, as well as on the topology. In addition, the factors surrounding node energy consumption and expected node lifetime further increases the complexity of the analytical approach. Alternatively, the deployment of real-life test-beds requires a huge effort, where a great deal of time can be lost troubleshooting problems that do not relate to the actual WSN design [Simulation tools for wireless sensor networks]. The complexity of the analytical approach and the huge deal of effort required for test-bed deployments make simulation an essential tool to study WSN. Use of simulators to perform WSN experiments has been described in many research papers, and is considered a valid method of evaluation.

4.1.1 Discrete Event Simulation

Discrete Event Simulation refers to modeling a system as a series of events occuring at discrete points in time, where the result of each event incurs changes in the overall state of the system. In a discrete-event simulator, events are recorded as event notices in a *future event list* (FEL) or *future event set* (FES), where each event is executed chronologically by the simulator. The occurence of an event may both trigger changes in the system state as well as generate new events to be added to the FEL/FES [24].

In the context of LLNs and WSN, discrete event-driven network simulators are widely used as a method of evaulation since they can be used as a flexible and inexpensive tool for testing the network without the need of having to deploy an actual physical network. They can also be more accurate and realistic than mathematical models where simplifications and abstractions are often assumed. Open-source simulators such as NS-2 and JSim have traditionally been used for simulating low-power wireless networks. Currently, NS-3, OMNet++ and TOSSIM are among the most widely used [25].

4.1.2 Choosing a simulator

A critical tasks in the planning of the project was selecting a suitable simulator. As there are many simulation tools available for networking and WSN, the feature set, workflow and available simulation models differ greatly. Learning how to use a particular simulator requires a considerable time investment, and this time will be wasted if the selected simulator turns out not suitable for the task at hand. Therefore, careful consideration of which tool to use is paramount.

For choosing a simulator, it is important to decide on a set of requirements which should be met by the chosen tool, such as: Are the required simulation models available? Is it easily usable and extensible? Is it still maintained and recently updated? Is it well-known, and has it seen any prior academic use? Most of these criteria should be suitably met before commiting to a certain tool. To this purpose, a evaluation of the most commonly used network simulators was performed in the beginning phase of the project. A number of simulators was considered, primarily evaulated on the basis of available models for TSCH and 6TiSCH, and secondarily on the aspects of previous academic use, extensibility, usability and recent updates. A summary of the evaluated simulation tools is given below:

OMNET++[26] was considered because of its reputation of being a "tried and true" simulator with a sizable community behind it. It uses C++ as well as its own modeling language NED to create and define simulation models. It also features an IDE used to manage and run simulations. Running natively in Windows, it offers a advantage over many tools that can only be run in an Linux-environment. However, currently OMNET++ lacks any models for simulating TSCH and/or 6TiSCH LLNs, which makes it unsuitable for the goal of this thesis.

NS3[27] was considered next, in a similar regard to OMNeT++ for being a well-known simulator backed by a big community. Like OMNeT++, NS3 is a C++ oriented simulator, but differs in the fact that it does not feature an IDE and must run in a Linux-environment. Models for evaulating TSCH LLNs does exist for NS3, however as of the time of writing, these models are not being currently maintained, and are also not part of the NS3 distribution. It is unclear if these models has been used in any publication, and it is thus difficult to ascertain their validity for academic use.

The *IRC-SPHERE TSCH-Simulator*[28] is a easy-to-use simulator designed for modeling simple TSCH networks. It is written in Python, and has been used to evaluate TSCH for applications in healthcare. While it offers many features compared to its relative low complexity, it is limited by the fact that it can only simulate star-topology networks, and does not feature multi-hop capability.

The 6TiSCH Simulator[25] is a simulator written in Python that focuses specifically on simulating 6TiSCH networks. It was created as part of the standardization activity by the IETF 6TiSCH WG, and as such closely follows the requirements specified in the 6TiSCH RFCs. It differs from OMNeT++ and NS3 in that is not a generic purpose simulator, but a specialized protocol specific tool for fast prototyping. [Simulating-6TiSCH] highlights features such as good scalability and low complexity as its key selling points. The 6TiSCH simulator is currently in active development, and has been utilized in several research papers.

Of all the simulators evaluated, the 6TiSCH Simulator comes closest to meeting the previously defined set of criteria about having models for simulating TSCH and/or 6TiSCH networks. It also scores highly when it comes to extensibility and usability, as a result of its low complexity, and being written completely in the Python languge. These points, along with the fact that it is currently being maintained and has seen use in papers, makes it an ideal choice for this thesis.

4.2 The 6TiSCH Simulator

This section will detail The 6TiSCH Simulator, which is this thesis' chosen tool for simulating 6TiSCH networks. The simulator is described in [25] by Municio et. al., and the following information is gathered from this paper.

4.2.1 Introduction

The 6TiSCH simulator[ref] is a discrete-event simulator written in the Python programming language. The design of the 6TiSCH simulator minimizes typical simulation drawbacks by making careful abstractions specific to 6TiSCH. This means that, instead of simulating physical behavior, it focuses on simulating the network from the perspective of the MAC layer. This is achieved with two abstractions. First, time is quantized into TSCH slots, which means an event can only take place at the slot boundary. Second, the protocol messages are abstracted to only carry semantically relevant parameters which also means that exchanged messages are not byte-accurate.

Building upon these two abstractions, the simulator can accurately monitor:

- the behavior of a Scheduling Function (SF) in response to generated traffic (in frames / second)
- the behavior of the routing protocol in response to topological changes
- the behavior of 6P in response to MAC-layer drops
- the behavior of the application in response to scheduling, routing, and network stack configuration.

The internal architecture of the simulator is shown in Figure 4.1. The main component is Mote, which is where most of the 6TiSCH stack is implemented. Mote is configured by SimSettings, which contains various parameters that can set by users. Mote also generates metrics for SimStats and SimGUI and creates events that are scheduled and later processed by SimEngine. An example of such an event is packet transmission and reception, which is evaluated by the Propagation component according to the current network topology defined in the Topology component.

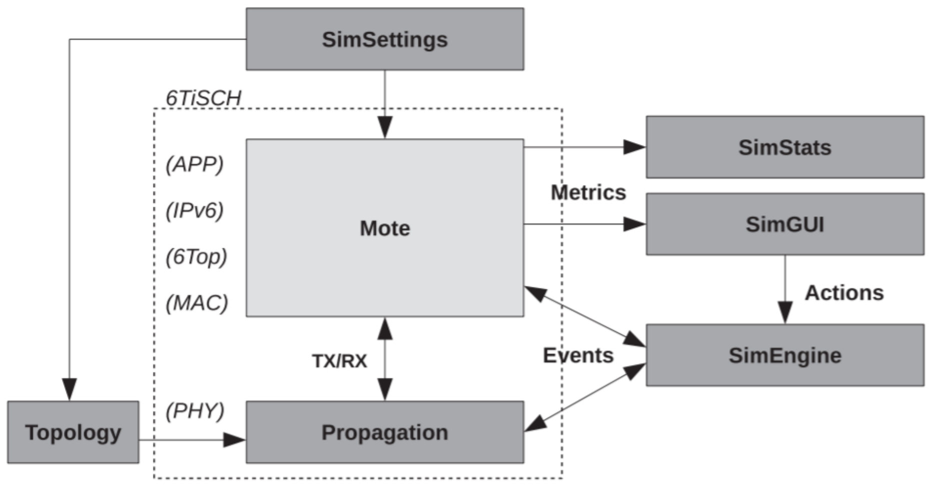


Figure 4.1: Architecture of the 6TiSCH Simulator, from [25]

4.2.2 SimEngine

The event-driven core of the simulator is implemented in SimEngine. Events are generated by the Mote every time a new task needs to be scheduled in the future, such as increasing the ASN in the nodes, propagating a packet, or firing a timeout. These events are uniquely identified by a tag formed by the node ID and a label, which can be considered as the event's universally unique identifier (UUID). Since more than one event can happen at the same time, events are registered with a specific priority and processed accordingly.

The set of events to be executed in the future, also known as the future event set (FES), is implemented using a Python list, in which events are added with the insert method and removed with the pop intermediate method. Figure 4.2 shows an example of how events are inserted and removed.

4.2.3 Topology

The simulator supports linear, random and fully-meshed topologies by default. In addition custom topologies are easily configurable. Network topology is defined by a connectivity matrix which indicates which motes are able to communicate which each other. Each cell of the connectivity

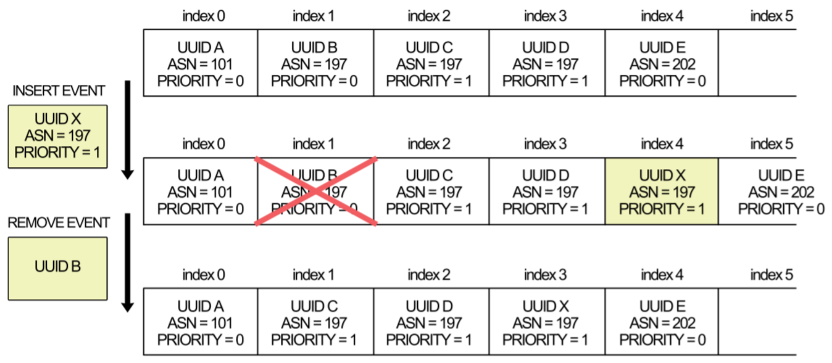

index 0
index 1
index 2
index 3
index 4
index 5
UUID A ASN = 101 PRIORITY = 0
UUID B ASN = 197 PRIORITY = 0
UUID C ASN = 197 PRIORITY = 1
UUID D ASN = 197 PRIORITY = 1
UUID E ASN = 202 PRIORITY = 0
INSERT EVENT
UUID X ASN = 197 PRIORITY = 1
REMOVE EVENT
UUID B

Figure 4.2: Example of how the future event set is managed in the 6TiSCH simulator. ASN, absolute slot number; UUID, universally unique identifier, from [25]

matrix defines a unidirectional wireless link between two motes, where connection parameters such as Recieved Signal Strength Indication (RSSI) and Packet Delivery Ratio (PDR) can be set.

4.2.4 Propagation Model

The default propagation model implemented in the 6TiSCH simulator is based on the Pister-Hack model [pister], which reflects the relationship between RSSI and PDR in large indoors industrial scenarios using the 2.4 GHz band. The Pister-Hack model and the RSSI-PDR conversion table are depicted in Figure 4.3 and Table 4.1, respectively.

The PDR values are used to calculate if a transmission is successful or not by flipping a biased coin. In order to model interference, the RSSI from the interfering neighbors is added to the ground noise in order to calculate the signal-to-interference-plus-noise ratio, which is then converted into the actual perceived PDR.

4.2.5 Energy Consumption Model

The simulator's energy consumption model is based on the model published by Vilajosana et. al. [29]. The model takes a component-based approach by defining the energy consumption of the different types of slots and combining them according to the schedule configuration. Figure 4.4 presents the sequence of actions that occur during a TSCH timeslot. This sequence of actions is used to model the energy consumed by a node.

The model considers different types of slots as per Table 4.2. After the execution of a slot, the simulator aggregates the consumed energy. By default, the simulator uses the OpenMote[30] platform as a reference for the current draw of each operation, but the values can easily be adapted to other platforms.

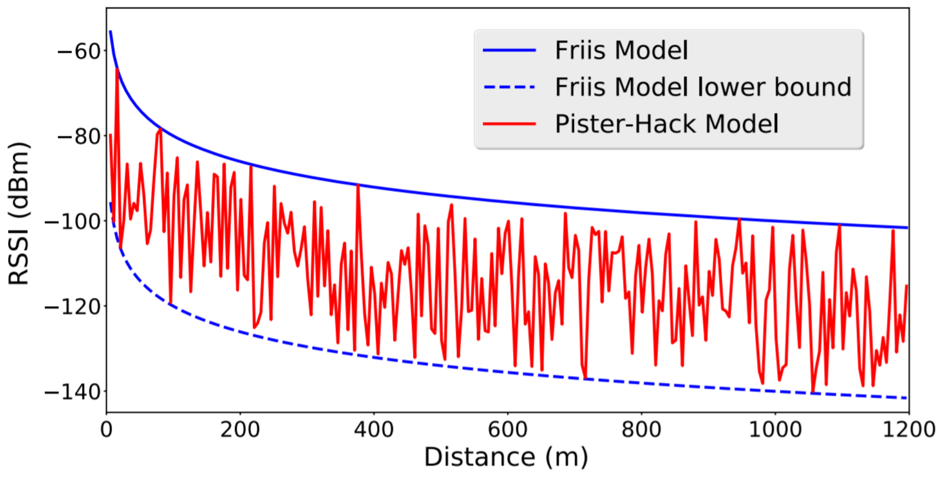


Figure 4.3: Received signal strength indicator (RSSI) values generated using the Pister-Hack model, from [25]

RSSI	PDR
-97 dBm	0.0000
-96 dBm	0.1494
-95 dBm	0.2340
-94 dBm	0.4071
-93 dBm	0.6359
-92 dBm	0.6866
-91 dBm	0.7476
-90 dBm	0.8603
-89 dBm	0.8702
-88 dBm	0.9324
-87 dBm	0.9427
-86 dBm	0.9562
-85 dBm	0.9611
-84 dBm	0.9739
-83 dBm	0.9745
-82 dBm	0.9844
-81 dBm	0.9954
-80 dBm	0.9903
-79 dBm	1.0000

Table 4.1: RSSI-PDR translation, adapted from [25]

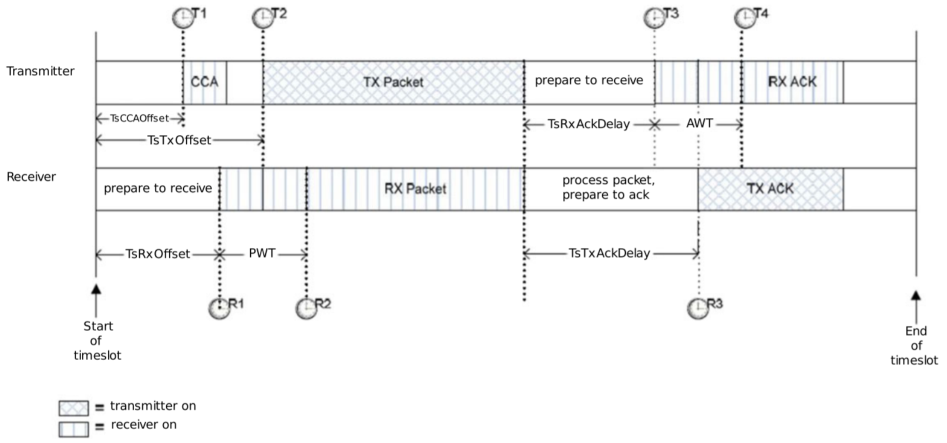


Figure 4.4: Timeslot timing and sequence of actions used to derive the energy consumption of a node, from [25]

State	Description
Idle	The node idle listens. This is an RX state where nothing is received. Hence, it only listens for the duration of the guard time.
Sleep	The node deep sleeps. The slot is off so no CPU nor radio activity due to communication.
TxDataRx-Ack	The node transmits a packet and receives an ACK for it.
TxData	The node sends a broadcast packet not requiring ACK.
RxData-TxAck	The node receives a packet and responds with an ACK.
RxData	The node receives a frame that does not require to be acknowledged.

Table 4.2: Possible slot types as per the energy consumption model implemented within the 6TiSCH simulator, adapted from [25]

4.2.6 6TiSCH Mote Model

In the 6TiSCH simulator, a "mote" is an abstraction of a 6TiSCH network node. The mote implements its different sublayers and core mote logic in the Mote file. The specified number of motes are instantiated at boot time, where the first mote (mote 0) is selected as root. The root is responsible for triggering network formation by periodically adding Enhanced beacons (EBs) and RPL Destination Information Objects (DIOs) to its transmission queue.

The remaining motes start listening on randomly picked channels in order to intercept EBs from the root. The propagation model will decide whether motes are able to detect and correctly receive EBs. When a node receives an EB, it synchronizes (enables its TSCH MAC layer) and starts the joining process. After joining the network through a join proxy, the node is able to decipher DIO messages and select a preferred parent and obtain a RPL rank. Following this, the node allocates dedicated cells to neighbors using the specified Scheduling Function.

Nodes schedule TX/RX events according to their schedule. At every slot, the Propagation model evaluates which nodes have scheduled transmissions at this ASN and which nodes are listening. For every packet, it will determine the outcome of the transmission regarding the signal strength and the interference level provoked by concurrent transmissions in the same radio vicinity.

The sublayers of the motes are highly configurable through the SimSettings component. It is possible to configure TSCH-related parameters such as slotframe size, timeslot duration, beacon period, etc. The 6top sublayer and the MSF scheduling function can be configured by changing parameters such as MAX_NUMCELLS, LIM_NUMCELLSUSED_LOW, HOUSEKEEPINGCOLLISION_PERIOD, etc. The RPL layer is implemented in the non-storing mode and supports configuring parameters such as periods for DIO and DAO messages. Finally, application traffic can be constant or variable and can be injected at any time during the simulation. Variable traffic can be modeled according to different probability distributions and configurable traffic bursts can be scheduled.

4.2.7 Metrics

During simulation execution, various event handlers trigger updates for the different metrics. Over 50 different metrics are currently implemented and the addition of new metrics is supported and is easily implementable. Metrics can be defined as per cycle to monitor the evolution of a specific metric per TSCH slotframe cycle or as absolute in order to obtain the resulting total value after the simulation. The slotframe cycle is the collection of timeslots in a slotframe and is set to 101 slots by default. Absolute metrics (such as charge consumed) can be obtained per node or aggregated over all the nodes in the network. RSSI values for every physical link are also logged. Table 4.3 details some of the most important metrics of the simulator.

The simulator by default logs the metrics of each simulation independently in a log file. Simulation runs use different files and folders depending on the CPU ID and the network size. The logging directory structure can be easily changed by the user in SimSettings. The simulator also provides with a set of helper scripts that allow the user to postprocess and plot any desired metric in a fully automated manner. An important part of postprocessing is the “compute_kpis.py” script which is responsible for creating files containing key performance indicators (KPIs) from the generated simulator log file.

4.2.8 Validation and previous academic use

The 6TiSCH simulator has been validated against OpenWSN, which is considered by [6TiSCH-simulator] as the most up-to-date 6TiSCH stack implementation available. It has also seen use in papers in various areas of research relating to both TSCH and 6TiSCH [31–34]

State	Type	Description
Average latency	Per cycle	Average latency of packets arriving at the root (in ASNs)
Charge consumed	Absolute	Charge consumed by all nodes during the simulation
Charge consumed at every node	Absolute	Total charge consumed by a node during the simulation
App packets generated/received	Per cycle	Number of data packets generated and received at every cycle
Number of TX/RX	Per cycle	Number of MAC frames sent and received at every cycle
Number of drops	Per cycle	Drops are classified by its cause: QueueFull, MaxRetries, and NoRoute
Number of used cells	Per cycle	TX/RX/SHARED cells used by all the nodes
Colliding cells	Per cycle	Dedicated cells used in more than one link
Parent changes	Per cycle	Number of parent changes per cycle

Table 4.3: Some of the available metrics in the 6TiSCH simulator, adapted from [25]

Chapter 5

Application of the 6TiSCH simulator

This chapter details the usage of the 6TiSCH simulator for this thesis, along with a description of the additions and modifications made to the simulator.

5.1 Additions and modifications

Several modifications and additions to the simulator was made in order to model a network with TRA. This includes the addition of a virtual Path Computation Entity (PCE), a TRA module, a traffic class for critical data, a number of custom parameters together with logging related to cell utilization. Modifications include adding support for tracks and the TRA-module at the TSCH layer, as well as tweaks to network convergence such as the ability to start the simulation with all nodes already TSCH-synced over a pre-configured static RPL dodag. Scripts for generating key performance indicator (KPI) values were also modified to include metrics associated with cell use.

5.1.1 Helper scripts

A few helper scripts were created in order to compute average values of the individual simulator runs, and present them in a comma separated value (CSV) table. Each column of the csv table contains values of a metric mapped to a specific X-value e.g. PDR, packet rate and TRA-type. The tables were used to plot graphs for various metrics and analyse data.

5.1.2 Path Computation Entity Module

The Patch Computation Entity module "`pce.py`" defines a virtual PCE that manages the creation of tracks. It contains the `_build_track_specified` method which when given a list of mote ids as its `motes` parameter will create a track along the listed motes with the specified `trackId` and `cellnum` amount of cells for each hop. This method is used in all the simulations to create tracks for critical data from source to sink. The track creation time is scheduled using the simulator event scheduling method `ScheduleAtAsn`:

```
def _schedule_track_building(self):
    self.engine.scheduleAtAsn(
        asn = self.settings.exec_startSend * 101,
        cb  = self._buildOneTrack,
        uniqueTag = 'build_track',
        intraSlotOrder = d.INTRASLOTORDER_STARTSLOT
    )
```

where `self.settings.exec_startSend` is the value representing the slotframe to start sending data. Track creation is instant and makes no attempt at simulating a realistic PCE where adding track cells involves back and forth transanctions between the PCE and the nodes involved in the track.

The PCE will create cells sequentially starting from timeslot 1 along the slotframe, forming a "daisy-chain" where overprovisioned cells for each hop are added directly following each other. The channel assignment is random between 0 and the number of available channels. Figure 5.1 provides an example of the resulting 6TiSCH schedule for a track created along a linear network of four motes (3 hops), with two cells per hop.

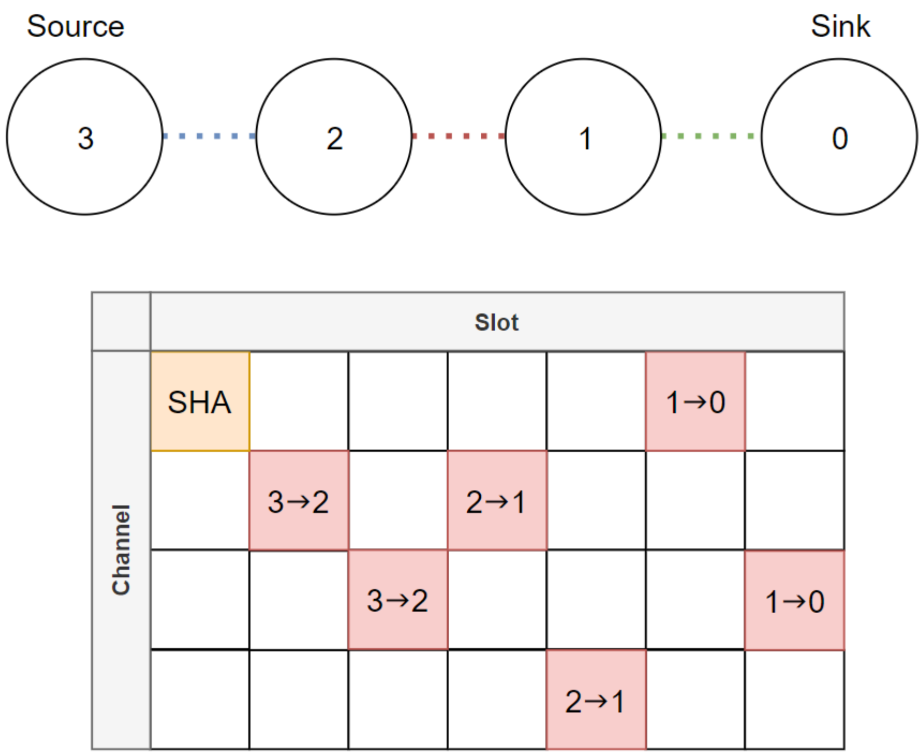


Figure 5.1: An example of a 6TiSCH schedule featuring a track created by the virtual PCE. SHA is the 6TiSCH minimal shared cell

5.1.3 TRA Module

The TRA module "`trae.py`" contains the logic for the proposed track resource adaptation mechanisms. It defines the base TRA-class `TRAEBase` along with subclasses `TRAEAllListen` and `TRAEOneShot`, representing the two proposed TRA-variants. The `TRAEBase` class can be used on its own to represent a network without TRA, as it only defines common TRA-related methods without implementing any functionality. For each simulation, all motes instantiate a specific TRA - variant from the classes defined in this module.

The module is dependent on methods called from the TSCH-module. These methods are illustrated in Figure 5.2 and allow the module to be notified about elapsed track cells and whether idle listening / packet reception occured, newly added track cells, and the state of the pending bit of received packets. A brief explanation of the TRA-module methods follows:

- `register_track_cell`, called whenever a TSCH-cell with a set `trackId` is added. Registers the cell with the TRA-module.
- `indicate_rx_cell_elapsed`, called whenever a track RX-cell has elapsed. The elapsed cell is passed as an argument.
- `indicate_active_rx_cell`, called when a RX-cell receives a packet. The cell and the packet pending bit is passed as arguments.
- `indicate_idle_rx_cell`, called whenever idle listening occurs on a track cell. The cell is included as an argument.
- `indicate_tx_cell_elapsed`, called when a track TX-cell has elapsed. The elapsed cell is included as an argument.
- `indicate_acked_tx`, called whenever a track TX-cell succesfully completes a transmission. The cell and the pending bit of the sent packet is included as arguments.
- `indicate_non_acked_tx`, called whenever a track TX-cell receives no ACK for a transmission. The cell is included as an argument.

Enabling and disabling cells

The module contains logic that allows it to decide whether to enable or disable track cells based on the state of the pending bit. This logic is contained in `indicate_active_rx_cell`, which is called whenever a cell receives a packet. The `pending_bit` argument reflects the received packets pending bit value, and is used to enable or disable the track RX-cells following this cell.

In order to support tracks with multiple daisy chains, as illustrated in Figure 5.3, the state of the last received pending bit is stored in the module. This ensures that even though a mote in a chain only has a single tracked packet in queue, the state of the stored pending bit will still be applied to the packet on TX to the next hop.

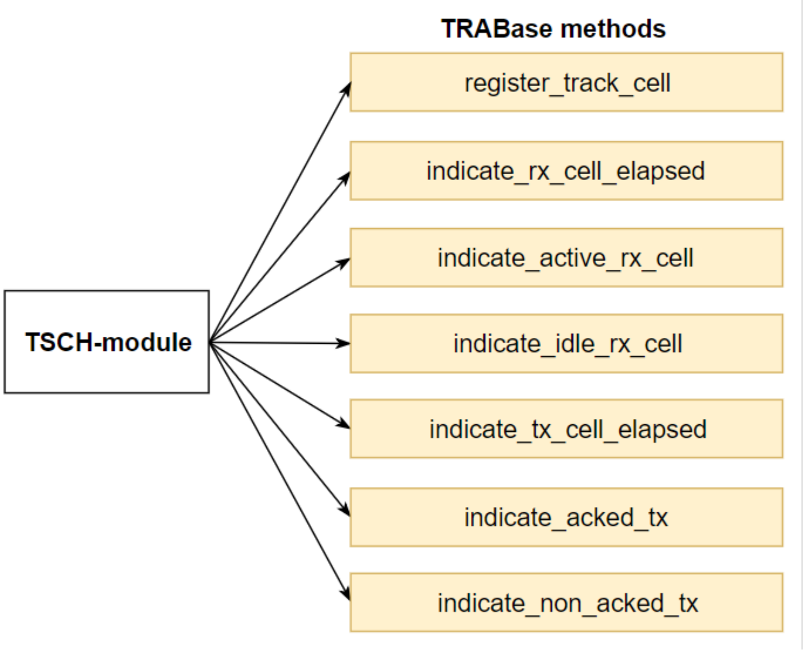


Figure 5.2: Overview of the TRA methods called from the TSCH-module

5.1.4 TSCH module modifications

A number of modifications to the simulator's native TSCH-module and its cell subclass was required. Since the 6TiSCH Simulator does not include the concept of tracks out-of-the-box, this feature was implemented by adding a `trackId` field to the module's `Cell`-class. All cells with the same `trackId` belong to the same track, and cells with a `trackId` of `None` are considered as not part of any track. Following this, a QoS mechanism was implemented that priorities critical packets on tracked cells. Support for TRA was added by ensuring that for any event (RX, TX, elapsed, cell creation) occuring on a track cell, the related TRA-module method should be called.

The ability to mark a cell as disabled was added; Disabled cells will remain allocated in the slotframe, but will not be usable for RX or TX, preventing any activation of the radio. As TSCH iterates through the slotframe it will sleep for any cells marked as disabled.

For a cell, individual counters for the amount of slots where the cell was idle listening, disabled and elapsed was also added.

Finally logic for when to set the pending bit for tracked packets was added.

Traffic and track cells

Only application traffic is allowed on tracked cells. This is done to only observe the impact of controlled app-related packet flows on TRA, eliminating the noise of RPL packets and. Non-app traffic is only allowed on shared cells or cells added by the SF.

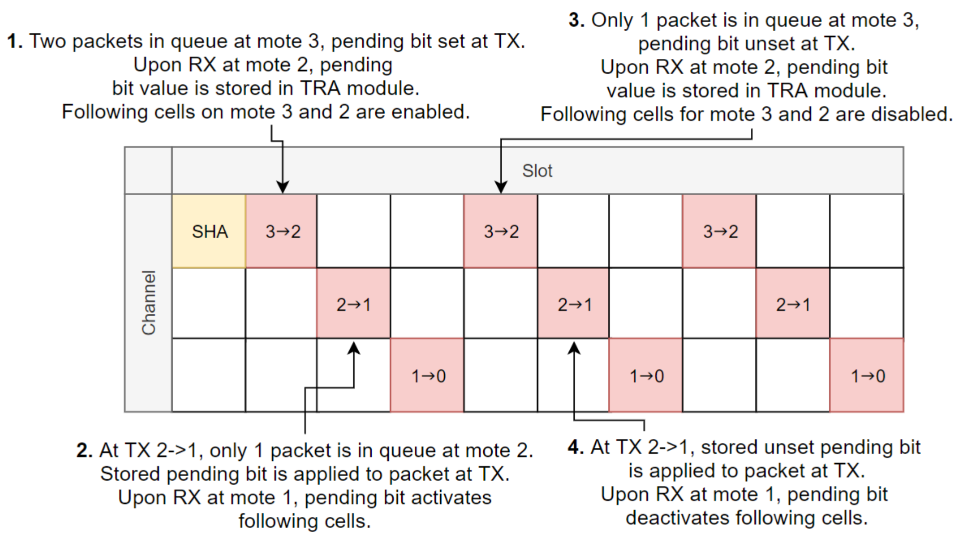

1. Two packets in queue at mote 3, pending bit set at TX. Upon RX at mote 2, pending bit value is stored in TRA module. Following cells on mote 3 and 2 are enabled.
3. Only 1 packet is in queue at mote 3, pending bit unset at TX. Upon RX at mote 2, pending bit value is stored in TRA module. Following cells for mote 3 and 2 are disabled.
Slot
Channel
SHA
3→2
2→1
1→0
2. At TX 2->1, only 1 packet is in queue at mote 2. Stored pending bit is applied to packet at TX. Upon RX at mote 1, pending bit activates following cells.
4. At TX 2->1, stored unset pending bit is applied to packet at TX. Upon RX at mote 1, pending bit deactivates following cells.

Figure 5.3: TRA with multiple daisy chains

5.1.5 Custom Mote Applications

Two custom applications, refered to in the simulator as *Apps*, were defined in the simulator's App-module. These apps are responsible for the generation of data packets towards the sink at various rates and different packet types. All packets generated have a payload length of 90 bytes. This value was chosen as it is the default value for the simulator, and also reprensents the maximum payload size of a packet before fragmentation occurs.

AppDataCriticalTracked

This app is responisble for the generation of critical packets towards the sink. The packets are generated in an "Alarm"-like fashion. For each timeslot / ASN, the app has a chance to generate N number of packets starting from slotframe iteration `exec_startSend`. Packet generation follows a Poisson distribution, where the parameter `crPktProb` represents λ which is defined as the amount of packets per slot.

```
for _ in range(0,
  ↪ poisson.rvs(self.settings.crPktProb)):
    # create data packet
```

AppPeriodicSlotFrame

Creates best effort traffic at a period equal to `beTxPeriod` slotframe iterations. For each period starting from slotframe iteration `exec_startSend`, a packet is generated at a random timeslot within the slotframe.

5.1.6 Custom parameters

In addition to the default parameters, custom parameters specifically related to various simulator additions were implemented, highlighted by Table 5.1.

Parameter	Description
pdr	Global packet delivery rate for links between nodes
num_track_cells_per_hop	Number of cells per hop in the PCE track
trae	Specified TRA method
exec_startSend	Slotframe to start sending application traffic
crPktProb	Critical data rate
beTxPeriod	Generation period in slotframes for best-effort traffic

Table 5.1: Custom simulation parameters

5.1.7 General modifications

General modifications to the simulator includes implementing metrics related to TRA such as counters for idle listening and adding a class for critical data, as well as additional logging of events. A big modification to the simulator was allowing motes to start TSCH-synced, and changing the RPL module to not allow parent changes that do not reflect the specified topology of the current simulation scenario. This was necessary to speed up and maintain convergence at low PDR values, but it must be noted that as a result an assumption is made that RPL is always able to maintain the desired topology.

5.2 Simulation metrics

Important simulation metrics are detailed in Table 5.2 They include a mixture of the default simulator metrics as well custom metrics related to total track cell usage per simulation run. The default metrics mainly relate to packets: e.g. latency and amount lost / received, while the custom metrics relate to track cell utilization. (Total) packets lost has been divided into specific metrics for both retransmission losses, and losses due to full transmission queues.

Metric	**Description**
Min, max, avg latency (s)	Latency for app data packets from generation to arrival at sink
Number of receive RX-cells	Total number of RX-cells that received a packet during simulation
Number of idle listen RX-cells	Total number of RX-cells that did not receive a packet during simulation
Number of disabled RX-cells	Total number of disabled RX-cells observed during simulation
Disabled RX-cell percentage	Percentage of disabled vs. enabled RX-cells observed during simulation
Idle listen ratio	Ratio of idle listening vs. packet receive events
Idle listening reduction (%)	Amount of reduction idle listening reduction compared to baseline
Packets lost	Total amount of packets dropped
Packets lost - Retransmission	Total amount of packets lost to retransmission drops
Packets lost - Queues full	Total amount of packets lost to full queues
Packets received	Number of packets received at sink
Network lifetime	Network lifetime in years

Table 5.2: Important simulation metrics

Chapter 6

Simulation scenarios

This chapter provides an overview of the simulation scenarios. A brief description of each scenario is given, along with details about implementation and simulator settings.

6.1 Common simulation parameters

This section will highlight parameters that remained constant throughout the simulations. Most of the parameters were left unchanged from the sample simulation configuration which follows some of the recommendations of IEEE 802.15.4 as well as the Minimal IPv6 Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) RFC [35]. Parameters relating to fragmentation are not listed as they are not considered relevant; all packets generated in the simulations have a payload length equal to `tsch_max_payload_length`, which prevents any fragmentation from occuring. As the two scenarios defined in the thesis don't rely on the simulator's default apps for packet generation, the parameters related to these apps are also not listed.

Parameter	**Value**	**Description**
numCPUs	1	Number of CPU cores to use for simulation
numRuns	30	Runs per parameter combiniation
exec_randomSeed	*random*	Random seed for simulation
secjoin_enabled	*true*	Use secure join process
rpl_daoPeriod	60 s	Period in seconds for sending DAO
tsch_max_payload_len	90 bytes	Max length of the packet payload before fragmentation occurs
sf_class	*MSF*	The used SF
tsch_slotDuration	10 ms	The length of a TSCH timeslot
tsch_slotframeLength	101	Number of timeslots in a slotframe
tsch_probBcast_ebProb	0.16	The probability for broadcasting an EB
tsch_clock_max_ drift_ppm	30	Amount of clock drift
tsch_clock_frequency	32768	Internal clock operation frequency
tsch_keep_alive _interval	10 s	Interval for sending keepalive messages
tsch_maxretries	5	Max number of TX retries per packet
phy_numChans	16	Number of channels available
tsch_queue_size	10	Size of the transmission queue at each mote

Table 6.1: The common simulation parameters

6.2 Scenario 1 - Critical data with 2 cells per hop

This scenario aims to investigate the impact TRA has on latency, packet loss and cell utilization under different PDR and packet generation rates for a linear network featuring a track with two cells per hop.

6.2.1 Description

The goal of TRA is to minimize the amount of idle listening occuring in 6TiSCH tracks featuring overprovisioned cells. Overprovisioning is the result of requiring retransmission opportunities in cases of unreliable wireless links, or the need to accomodate peaks for traffic with variable rates. This scenario is intended to evaluate the performance of TRA in networks where such overprovisioning is present. The performance of TRA is evaluated on the basis of how it compares to a standard 6TiSCH network without any TRA mechanisms.

6.2.2 Implementation

A linear network consisting of four motes is considered. Mote 3 is the source node and mote 0 is the sink. Mote 3 generates critical data following a Poisson-distribution with a λ of *crPktProb* packets per timeslot. This data is sent over 3 hops, using a 6TiSCH track created by the PCE-module. All motes along the track implement the same TRA-mechanism (e.g. All Listen, One-Shot). Each hop is made up of two cells in direct succession towards the neighbour mote, representing a case of overprovisioning. The Minimal Scheduling Function (MSF) is run as the underlying scheduling function for non-app traffic such as RPL messages, and is delegated a specific range of timeslots in the schedule as to not overlap with the track cells. Figure 6.1 provides an illustration of the scenario topology and TSCH-schedule.

6.2.3 Simulator settings

The simulations used the commonly defined simulator settings from Table 6.1. Scenario specific settings are shown in Table 6.2. Parameters `exec_startsend` and `exec_numSlotframes` were adjusted per PDR to allow the network to converge before generating data packets. Both were set so the source mote would transmit packets for 20000 slotframe iterations, which amounts to around 5 and a half hours with a slotframe length of 101 timeslots and 0.01 seconds per slot. PDRs in the range of 1.00 to 0.70 with 0.05 step decrements between each value was considered, along with the packet generation rates (λ) 0.001, 0.005 and 0.01. Each TRA-method was tested in conjunction with all possible parameter combinations. 30 simulations was performed per combination, resulting in a total of 1890 ($3 \cdot 3 \cdot 7 \cdot 30$) individual simulator runs. Results from each set of combination runs were averaged, and the min and max latency values were identified among the runs.

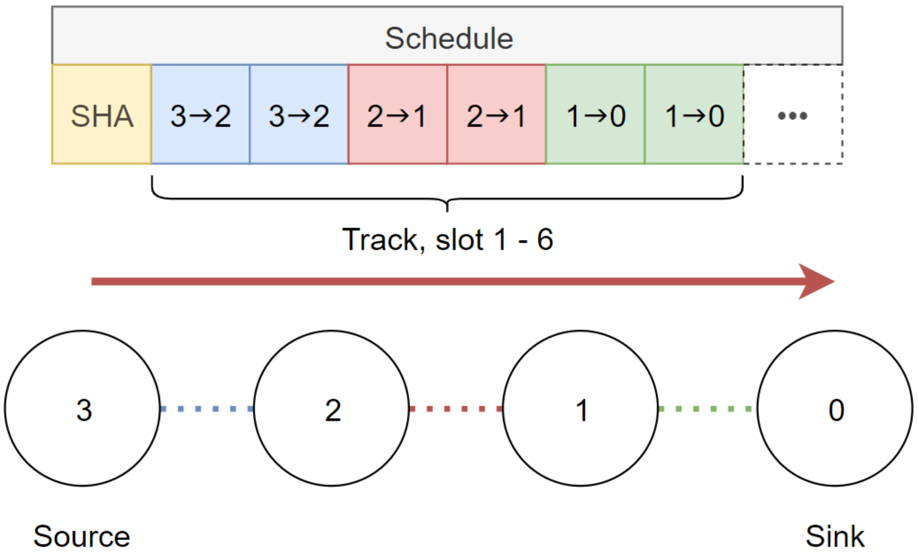


Figure 6.1: Scenario 1 topology and schedule

Parameter	**Value**
exec_numMotes	4
num_track_cells_per_hop	2
trae	None, All Listen, One-Shot
pdr	1.00, 0.95, 0.90, 0.85, 0.80, 0.75, 0.70
crPktProb	0.01, 0.005, 0.001

Table 6.2: Scenario-specific settings for Scenario 1

The specific combinations were chosen to investigate performance in applications which have different reasons for overprovisioning. At one end, with `crPktProb` of 0.001 and low PDRs, the combinations represent a case where overprovisioning could stem from allowing further retransmission attempts. At the other end, with `crPktProb` of 0.01 and high PDRs, the combinations represent a case where overprovisioning is due to allocating bandwidth for peaks in traffic.

Table 6.3 lists the expected amount of packets generated for each value of `crPktProb` (λ) following Eq 2.6.

crPktProb (λ)	**Expected packet count**
0.01	20 200
0.005	10 100
0.001	2 020

Table 6.3: Expected packet count for Scenario 1

6.3 Scenario 2 – Critical data with 3 cells per hop

This scenario increases the amount of cells per hop from Scenario 1 to 3 in order to investigate the impact of further overprovisioning on TRA.

6.3.1 Description

Similarily to Scenario 1, the aim of this scenario is to evaluate the effectiveness of TRA in a network with even further overprovisioning. Three cells per hop allows for more retransmission opportunities within a single slotframe, as well as greater bandwidth for transmission of peak traffic. It also allows a mote to more easily recover from congestion, preventing packet loss due to full queues. It also means that more energy is wasted on idle listening, which could be mitigated by a TRA mechanism.

6.3.2 Implementation

Similarly to scenario 1, a linear network with 4 motes is considered, where mote 3 is the source mote and mote 0 is the sink. Mote 3 again generates critical data following a Poisson-distribution with a λ of *crPktProb* packets per timeslot. The packets are sent to the sink over a track created by the PCE, this time featuring three cells per hop. All motes run the same TRA-mechanism, and MSF is used as the underlying scheduling function for non-app traffic.

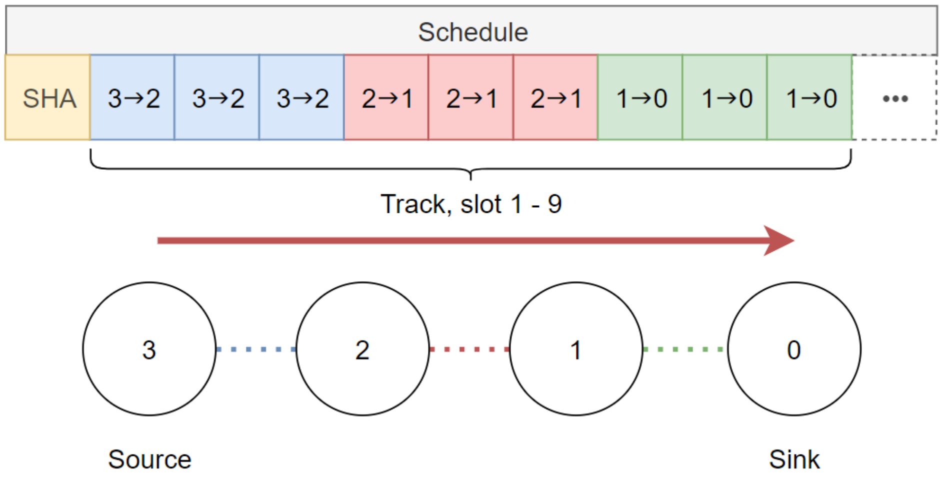


Figure 6.2: Scenario 2 topology and schedule

6.3.3 Simulator settings

The parameter settings for this scenario are equal to the settings of the previous, with the only change being the increase in `num_track_cells_per_hop`. Parameters `exec_startsend` and `exec_numSlotframes` were adjusted per PDR for convergence, and to allow the source mote to transmit packets for 20000 slotframe iterations.

Parameter	**Value**
exec_numMotes	4
num_track_cells_per_hop	3
trae	None, All Listen, One-Shot
pdr	1.00, 0.95, 0.90, 0.85, 0.80, 0.75, 0.70
crPktProb	0.01, 0.005, 0.001

Table 6.4: Scenario-specific settings for Scenario 2

6.4 Scenario 3 – Critical data and best-effort data

This scenario intends to investigate the impact track cell reuse has on TRA. Packet latency and loss and cell utilization are considered important metrics.

6.4.1 Description

The goal of this scenario is to determine what effect TRA has on cell reuse. Cell reuse is a feature of 6TiSCH tracks that allow packets not belonging to a track to make use of track cells if there is currently no track traffic available to send, and the next hop destination of the packets is the the same as for the track cell. The scenario intends to observe how TRA affects the latencies and reliability of critical and best-effort traffic as well as the impact on track cell utilization. In this scenario, critical data has priority on the track, while best-effort data does not.

6.4.2 Implementation

A network with five motes is considered. The scenario topology and TSCH-schedule is illustrated in Figure 6.3. The topology is partly based on a similar one presented by Theoleyre et. al. [36], in which data belonging to two applications is transported from two sources over a common track to a destination. In this scenario, Mote 3 and 4 are two data sources and mote 0 is the sink. Mote 3 generates critical data following a Poisson-distribution with a λ of *crPktProb* packets per timeslot, and mote 4 generates one best-effort packet every *beTxPeriod* in a random slot of the current slotframe. All data is sent to the sink. Motes 3 to 0 host a track for critical data towards the sink, and mote 4 has two non-track cells for traffic towards mote 2. Best-effort traffic is able to travel along the track, but critical data has priority. For any track TX-cell, if a mote has both a critical and best-effort packet in its queue, the critical packet will be chosen for TX. Best-effort packets will only be chosen for TX if there are no critical packets in queue. MSF is run as the underlying scheduling function for non-app traffic such as RPL messages, and is delegated a specific range of timeslots in the schedule to not overlap with the track cells.

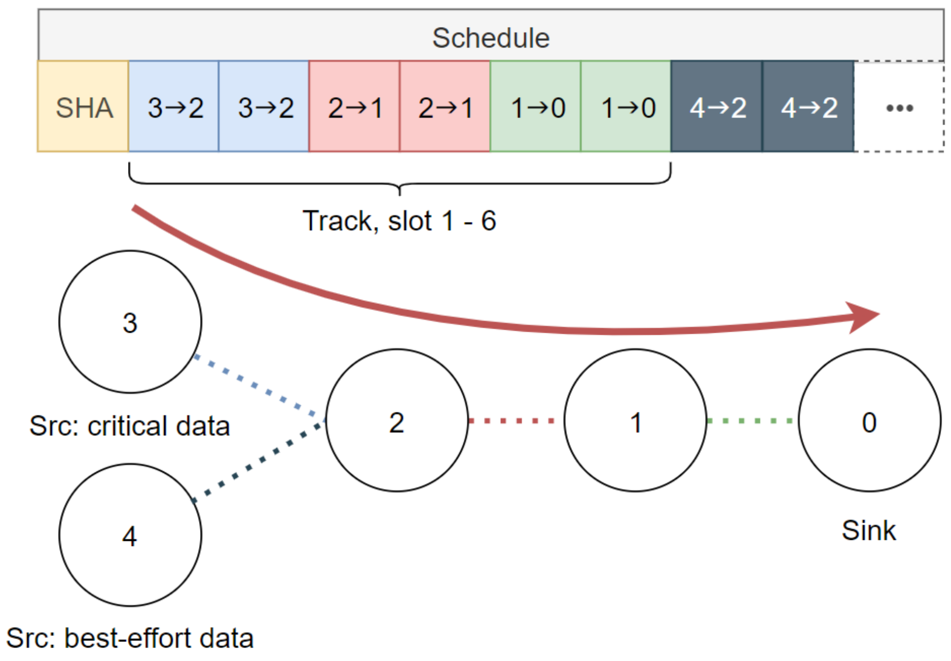


Figure 6.3: Scenario 3 topology and schedule

Parameter	**Value**
exec_numMotes	5
num_track_cells_per_hop	2
trae	None, All Listen, One-Shot
pdr	0.95
crPktProb	0.005
beTxPeriod	1, 2, 3 (slotframes)

Table 6.5: Scenario-specific settings for Scenario 3

6.4.3 Simulator settings

The simulations used the commonly defined simulator settings from Table 6.1. Scenario specific settings are shown in Table 6.5. Similarly to the previous scenario, `exec_startsend` and `exec_numSlotframes` were set so both packet generating motes would generate packets for 20000 slotframe iterations. The value of `crPktProb` remained at 0.005. Using Eq 2.6 this gives an expected total packet count of $P_{totalcr} = 101\,000$ for critical packets. `beTxPeriod` was set to 1, 2 and 3 to investigate the effect of different best-effort traffic rates. Table 6.6 lists the total expected best-effort packet count for each `beTxPeriod` as given by Eq 2.7.

beTxPeriod	Expected BE packet count
1	20 000
2	10 000
3	6 666

Table 6.6: Expected best effort packet count, Scenario 3

Chapter 7

Simulation results and analysis

This chapter will present simulation results and provide analysis for the three scenarios.

7.1 Scenario 1 - Critical data with 2 cells per hop

This section will detail the results of simulation scenario 1. For this scenario, a network with a 3-hop linear topology was considered, with a source mote in one end and a sink mote in the other. A track was created with 1 overprovisioned track cell per hop, resulting in a total of 2 track cells per hop. The simulations were run for each TRA variant with combinations of λ values 0.01, 0.005, 0.001, and PDRs 1.0, 0.95, 0.90, 0.85, 0.80, 0.75, 0.70. Packets were generated for 20000 slotframes per simulation, and metrics related to track cell usage was only recorded for these slotframes. All combinations were also performed on a series of non-TRA baseline simulation runs, refered to as "None", in order to compare the results to the current state-of-the-art. 30 runs were performed per parameter combination, and the results of all metrics were averaged.

7.1.1 Results

Results are split into latency and packet drops, idle listening reduction, and lifetime increase.

Latency and packet drops

Results for latency and average packet drops are illustrated in figures 7.1, 7.2 and 7.3 and are grouped by different values of λ.

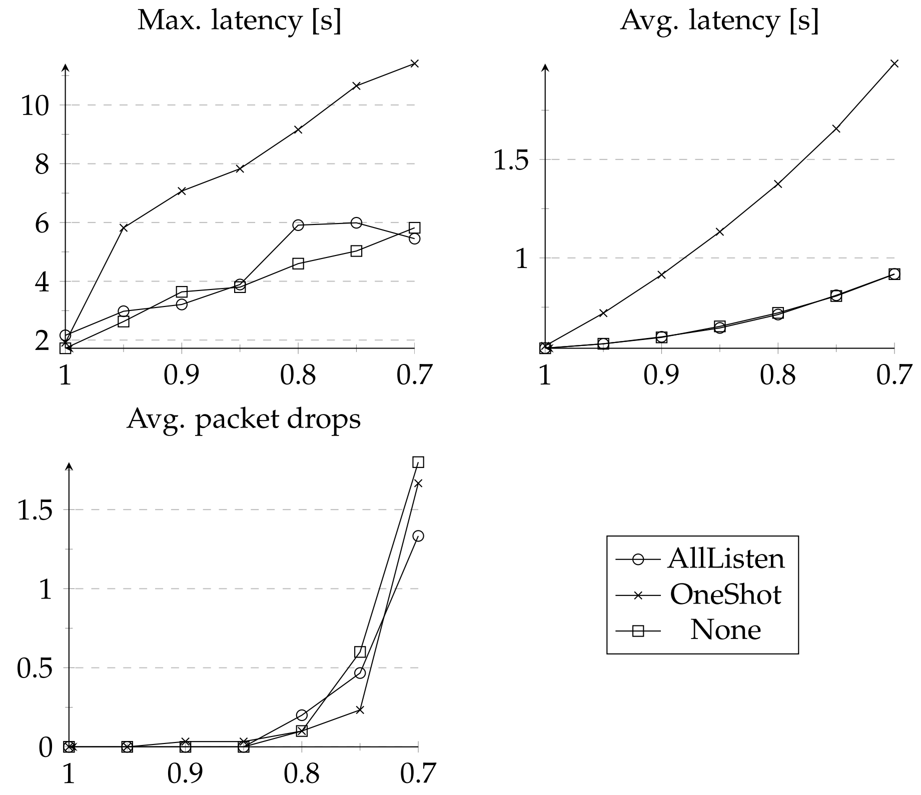


Figure 7.1: Latency and average packet drops vs PDR for Scenario 1 @ $\lambda = 0.001$

Results for $\lambda = 0.001$ are displayed in Figure 7.1. Latency and average packet drops increase as the PDR decreases. While AllListen and None perform similarly for all metrics, OneShot demonstrates higher values for max. and average latency. Average packet drops is similar for all variants.

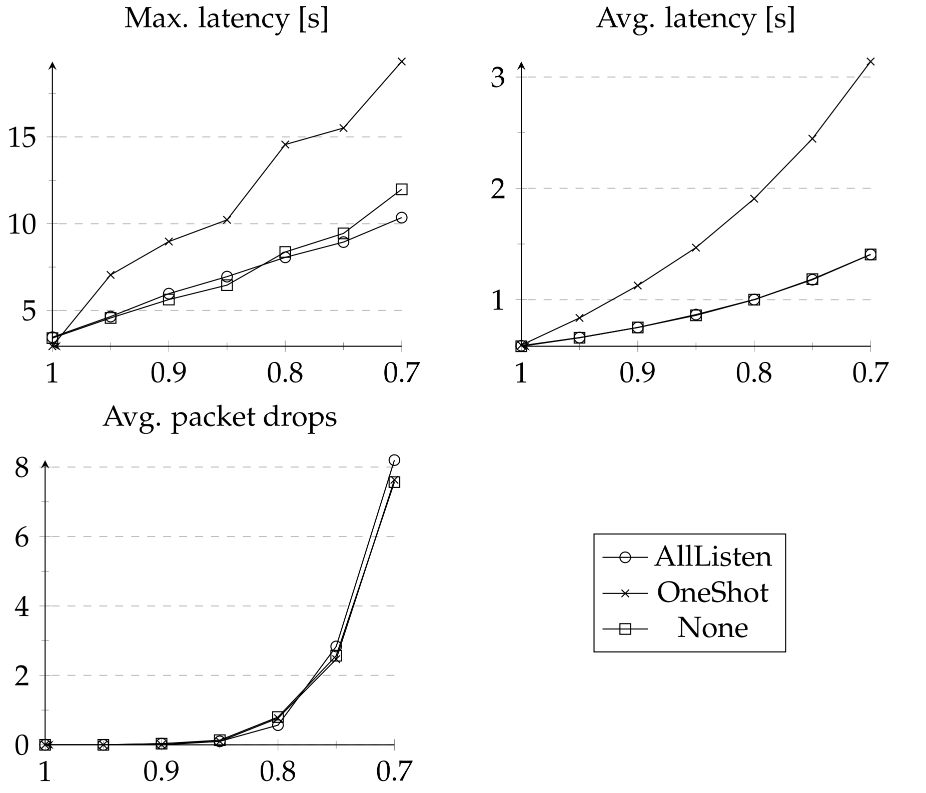


Figure 7.2: Latency and average packet drops vs PDR for Scenario 1 @ $\lambda = 0.005$

Figure 7.2 illustrates the results for $\lambda = 0.005$. As the packet rate increases, there is also an increase in latency and average packet drops. OneShot again deviates from AllListen and None by showing larger values for max and average latency as PDR lowers. Average packet drops is still similar for all variants.

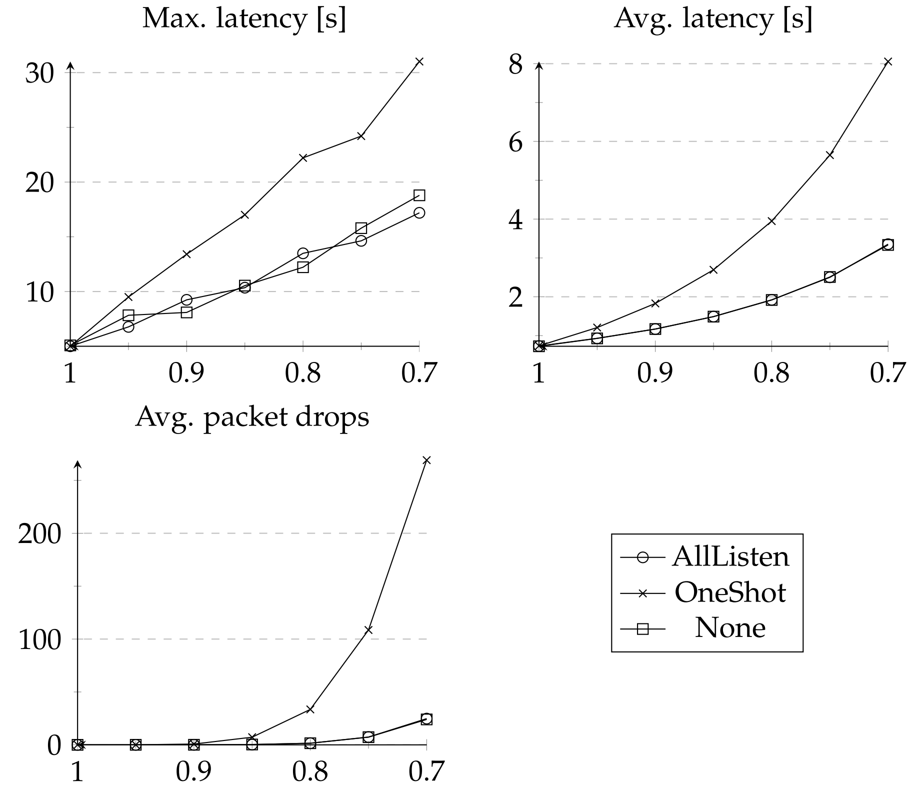
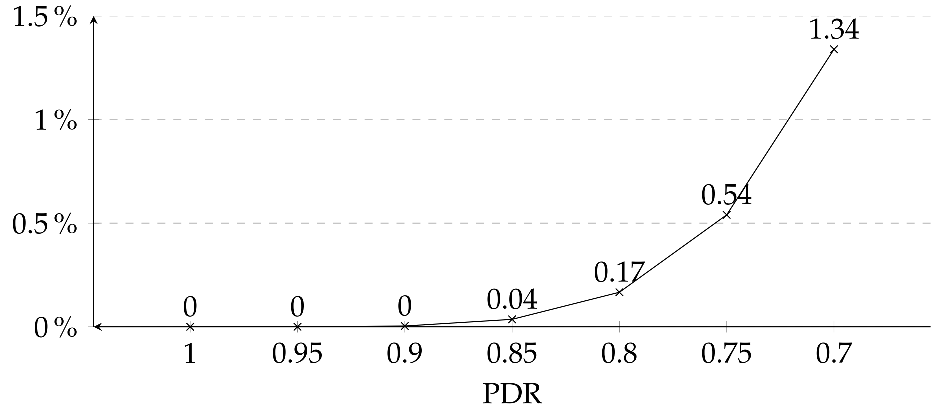
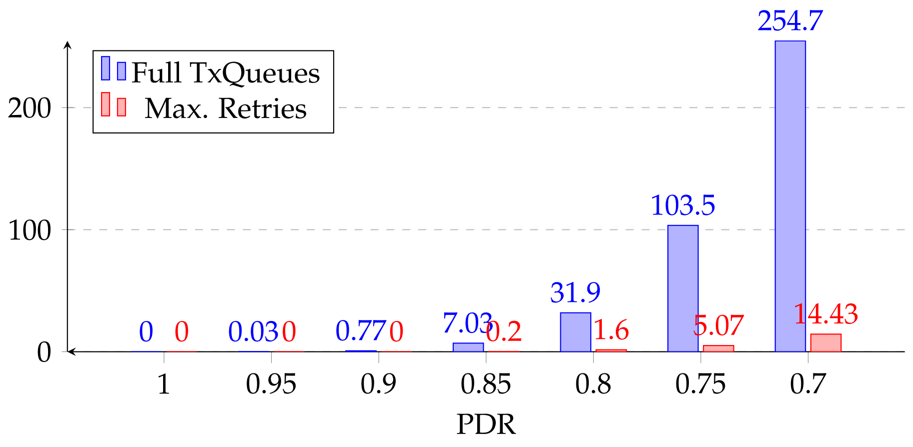


Figure 7.3: Latency and average packet drops vs PDR for Scenario 1 @ $\lambda = 0.01$

Figure 7.3 shows the results for $\lambda = 0.01$. There is again an increase in latency and average packet drops for all variants. OneShot still has the highest values for both max and average latency, and also suffers from higher packet loss as PDR goes below 0.85. AllListen and None closely match values for both latency and packet drops for all PDRs. Figure 7.4 shows packet loss in percent and drop reasons for OneShot at this packet rate.



(a)



(b)

Figure 7.4: a. Packet loss rate in percent and b. Drop reasons for OneShot, for $\lambda = 0.01$, Scenario 1

Idle listening reduction

Figures 7.5 and 7.6 presents reduction in idle listening vs PDR for track cells in the network. Results are presented individually for each TRA-variant, with plots for each packet rate (λ). Reduction is presented as the percentage decrease of total observed idle listening events compared to None.

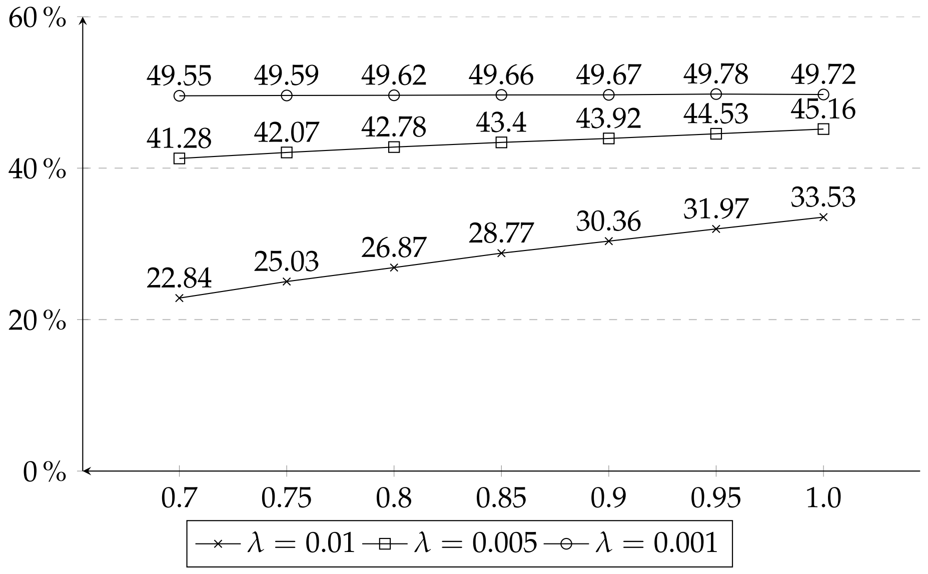


Figure 7.5: Idle listening reduction vs PDR for OneShot, Scenario 1

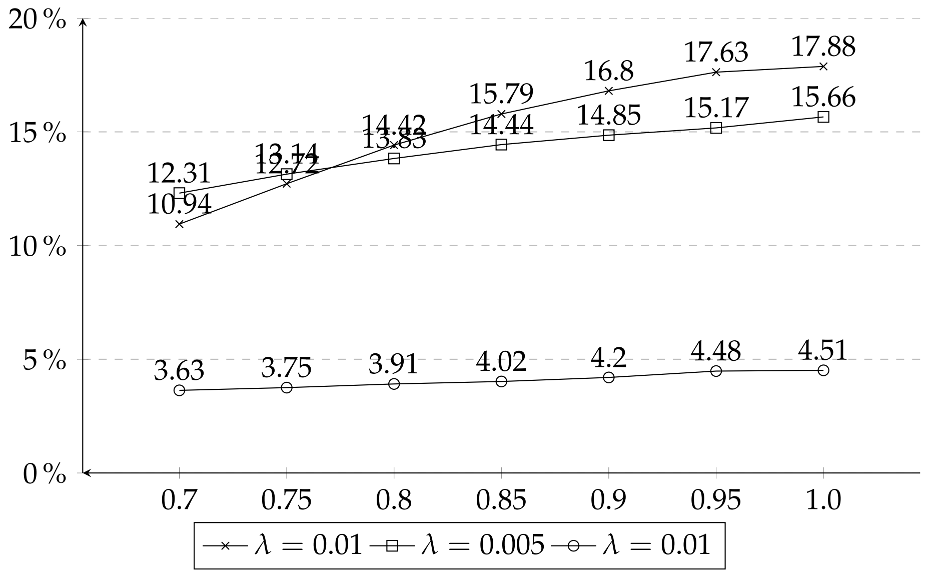


Figure 7.6: Idle listening reduction vs PDR for AllListen, Scenario 1

Network lifetime increase

Increase in network lifetime is illustrated by figures 7.7 and 7.8, and is presented individually for each TRA-variant with plots for each value of λ. The increase is reported as the difference in network lifetime in years compared to None.

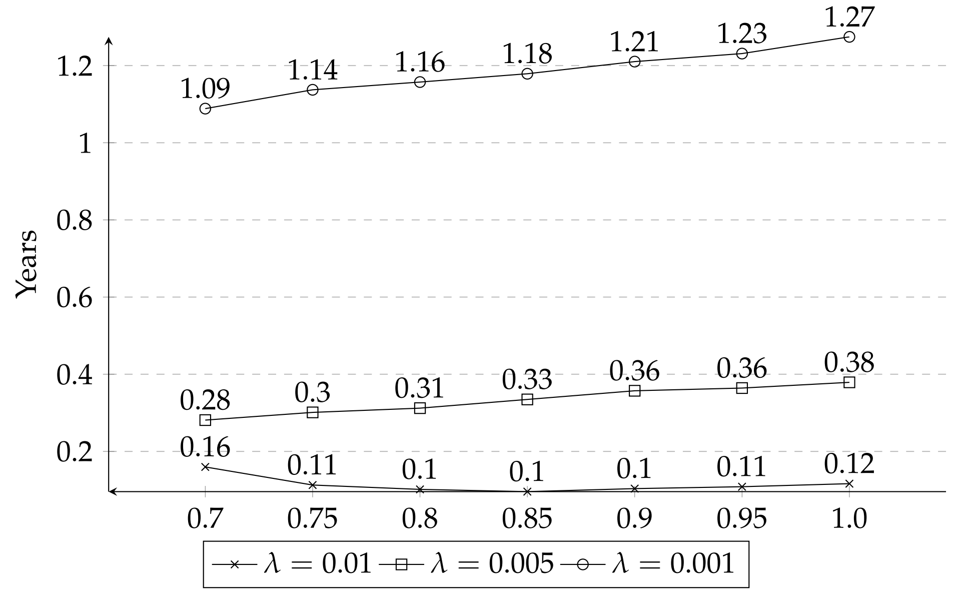


Figure 7.7: PDR vs increase in network lifetime for OneShot, Scenario 1

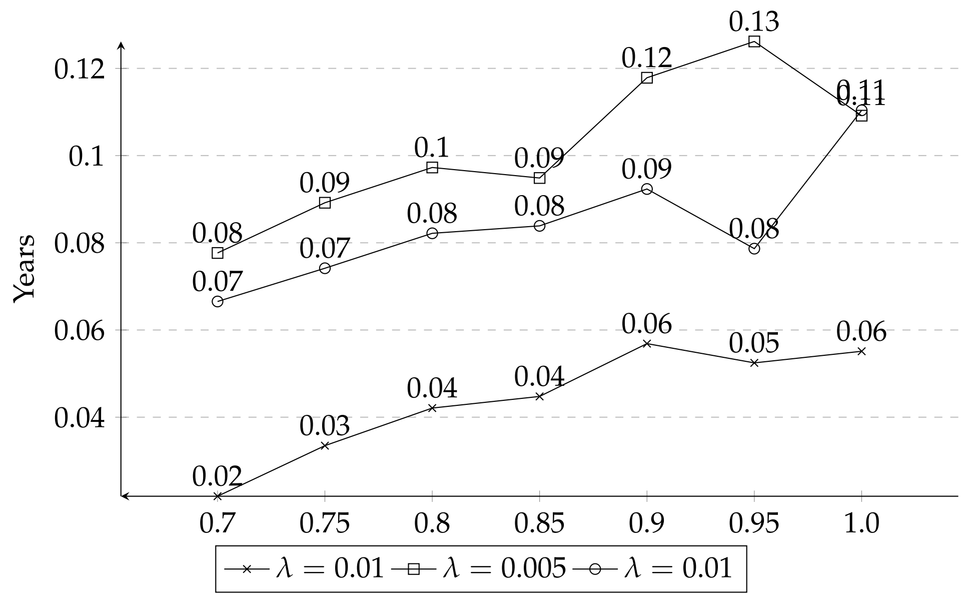


Figure 7.8: PDR vs increase in network lifetime for AllListen, Scenario 1

7.1.2 Analysis and discussion

There is a difference between the two TRA-variants apparent throughout all simulations. OneShot overall demonstrates a higher idle listening energy reduction, but suffers greatly when it comes to max and average latency as the PDR decreases. It also struggles when dealing with higher packet rates, as drops increase drastically for lower PDRs. As seen in Figure 7.4, at 0.7 PDR OneShot suffers from a loss of 1.3% of all traffic for rate $\lambda = 0.01$. The vast majority of packets lost stem from packet drops due to full TxQueues which would indicate that OneShot suffers heavily from congestion at lower PDRs. Congestion is caused by TxQueues filling at a faster rate than the mote is able to transmit packets, and would normally be mitigated by the overprovisioned cell present in the track, allowing more transmission opportunities within a slotframe. However, OneShot is dependent on a succesful transmission (ACK) of a packet with the pending bit set to activate the overprovisioned cell. During low PDRs, packets will fail more often, leaving the overprovisioned cell deactivated for the current slotframe. The effects of congestion unfavorably impacts the throughput of the track, and is the cause for the observed congestion loss.

The high max and average latency present in OneShot is also related to congestion. As queues fill, packets have to "wait in line" for TX, and with many packets queued up, each maybe requiring multiple retransmission attempts, packets might end up queued for a long period of time, causing max latency to reach up to 30s (or around 30 slotframes), as seen in Figure 7.3. It seems OneShot already begins seeing the effect of congestion starting from PDR 0.95, as evident by the increase in latency compared to the other variants.

While OneShot offers great idle listening reduction at the cost of increased latency and congestion loss, AllListen paints a much more modest picture. It is not able to provide the same amount of idle listening reduction, but does manage to closely match the average latency of the None baseline for all packet probabilities and PDRs, which could make it suitable for applications with stricter requirements for latency.

Regarding idle listening reduction, there is a slight decline in efficiency as the PDR lowers for both variants. This is because as PDR decreases, more retransmissions will occur, and less cells will be left idle listening. This is compounded by the packet rate, as more packets equals more retransmissions, leading to increased cell utilization. As such, possible energy gain from idle listening reduction will shrink as the PDR lowers.

Although the results indicate that AllListen offers better idle listening reduction as λ increases, it is important to remember that at higher packet rates, more energy will be spent transmitting packets, and as the energy consumed by RX/TX is much larger than the energy consumed by an idle listening cell, the energy saved will only make up a small percentage of the total energy usage. Figure 7.8 illustrates the reported increase in network lifetime for AllListen. It is clear that the network lifetime gain is the highest for $\lambda = 0.005$ and PDR=0.95 with an increase of 0.13 years, even though the idle listening reduction for this point is only 15.17% compared to 17.63% for $\lambda = 0.01$ as shown by Figure 7.6.

As stated earlier, OneShot performs quite poorly when facing high packet rates combined with low link quality. However, for the packet rates evaluated in this thesis, packet loss only seems to become apparent at PDRs below 0.90. Thus, OneShot could be viable in environments featuring such characteristics. Figure 7.7 demonstrates an impressive network lifetime increase for OneShot, especially for $\lambda = 0.001$ where the lifetime has been increased by a just over a year. It is though unlikely that an application with such low packet rates would require overprovisioning in the first place, unless the goal was reducing latency by allowing more retransmission attempts. In that case, as OneShot only activates its overprovisioned cells upon reading a set pending bit, essentially not allowing retransmission for a single packet, it is not suited for this scenario. In different scenarios, featuring burst of traffic, with high PDRs / good wireless links, the applicability of OneShot might be more reasonable, since overprovisioning would then be used as a means to accomodate worst-case traffic spikes. If we then consider a network with a sparse periodic packet rate, which occasionally is subject to bursts of traffic, OneShot could offer an advantage in being able to reduce the amount of idle listening happening during non-burst traffic, as long as the latency introduced can be accepted.

7.2 Scenario 2 - Critical data with 3 cells per hop

This section details the results of simulation scenario 2. The purpose of Scenario 2 is to investigate the effect of adding another overprovisioned track cell per hop to the network of scenario 1. As such, the topology now features 3 track cells per hop. The parameter combinations are the same as for the previous scenario.

7.2.1 Results

Reults are split into latency and packet drops, idle listening reduction, and network lifetime increase.

7.2.2 Latency and packet drops

Results regarding max / average latency and average packet drops are presented in figures 7.9, 7.10 and 7.11 and are grouped by different λ values.

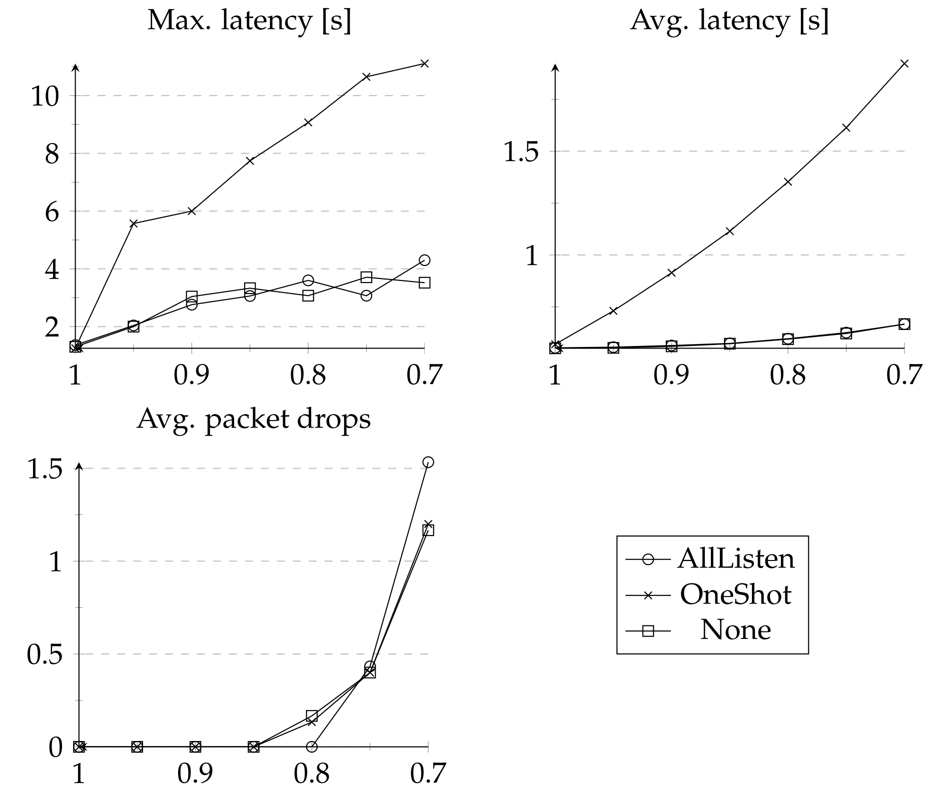


Figure 7.9: Latency and average packet drops vs PDR for Scenario 2 @ $\lambda = 0.001$

Figure 7.9 displays the results for $\lambda = 0.001$. Max and average latency is lower for AllListen and None compared to Scenario 1, while OneShot shows little difference. Average packet drops are the same as in the previous scenario for all variants.

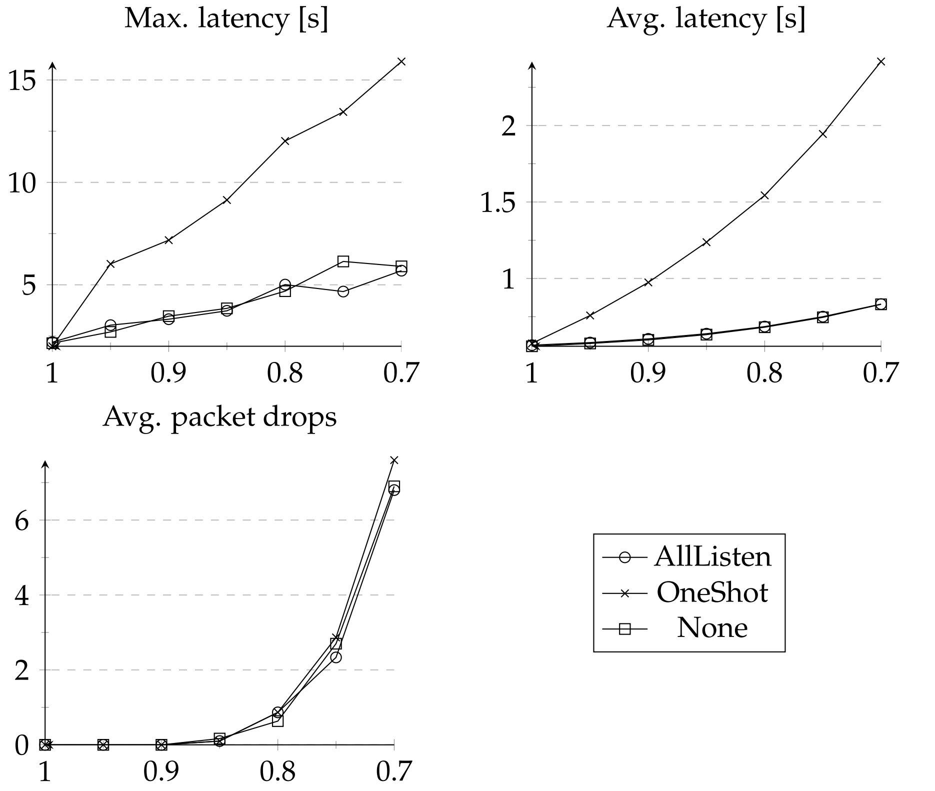


Figure 7.10: Latency and average packet drops vs PDR for Scenario 2 @ $\lambda = 0.005$

Figure 7.10 indicates the results for $\lambda = 0.005$. The values for max and average latency have decreased for all variants compared to Scenario 1. Average packet drops is also lower than in the previous scenario.

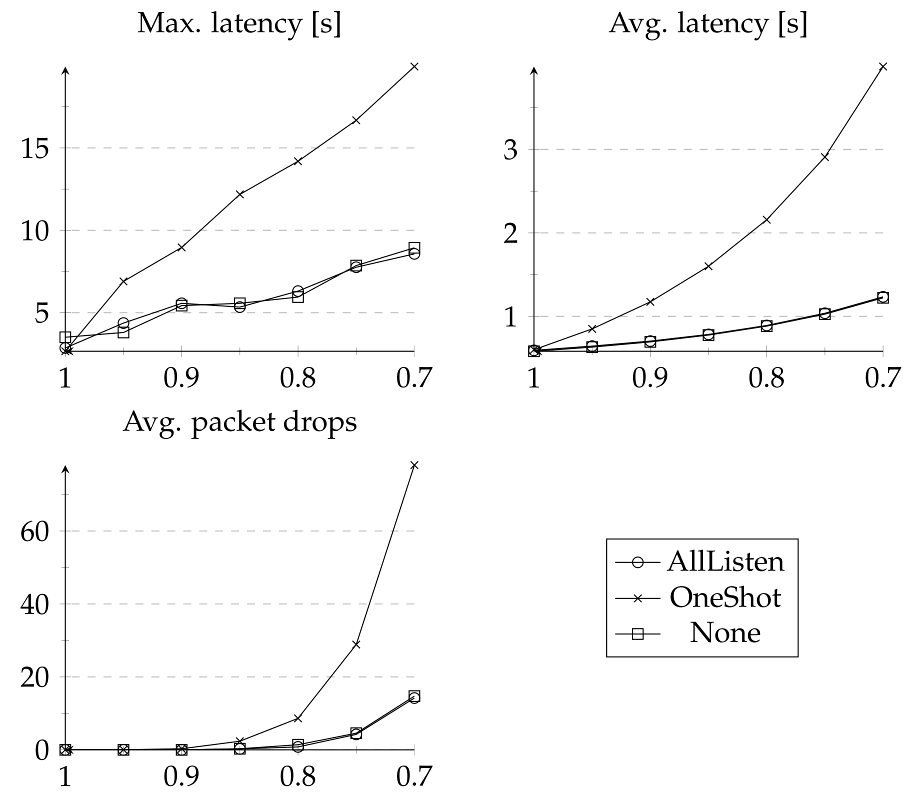


Figure 7.11: Latency and average packet drops vs PDR for Scenario 2 @ $\lambda = 0.01$

Figure 7.11 illustrates the results related to $\lambda = 0.001$. Latency and packet drops continue to be lower than in Scenario 1. Congestion loss is especially lower for OneShot compared to the previous scenario.

Idle listening reduction

Idle listening reduction is presented in figures 7.12 and 7.13. The amount of idle listening reduction is higher than in the previous scenario.

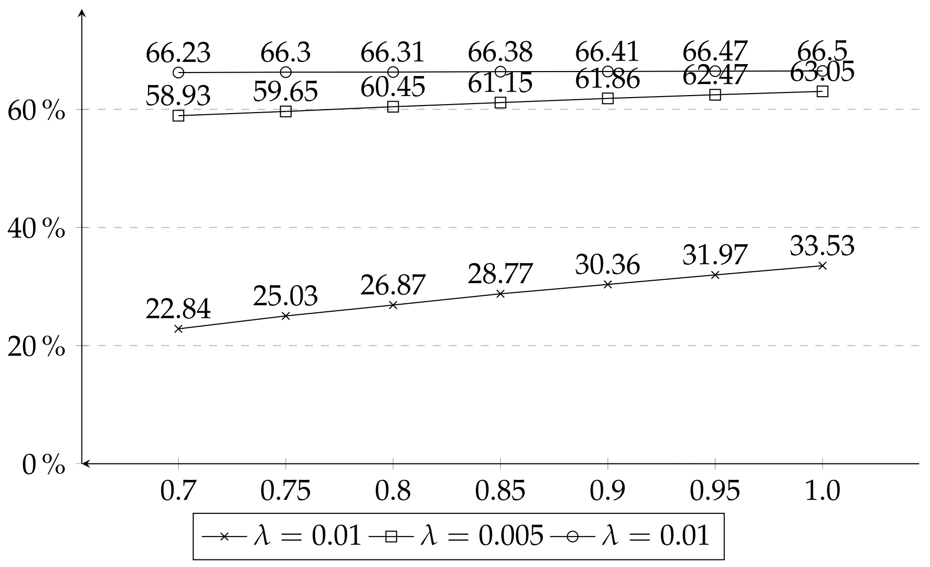


Figure 7.12: Idle listening reduction vs PDR for OneShot, Scenario 2

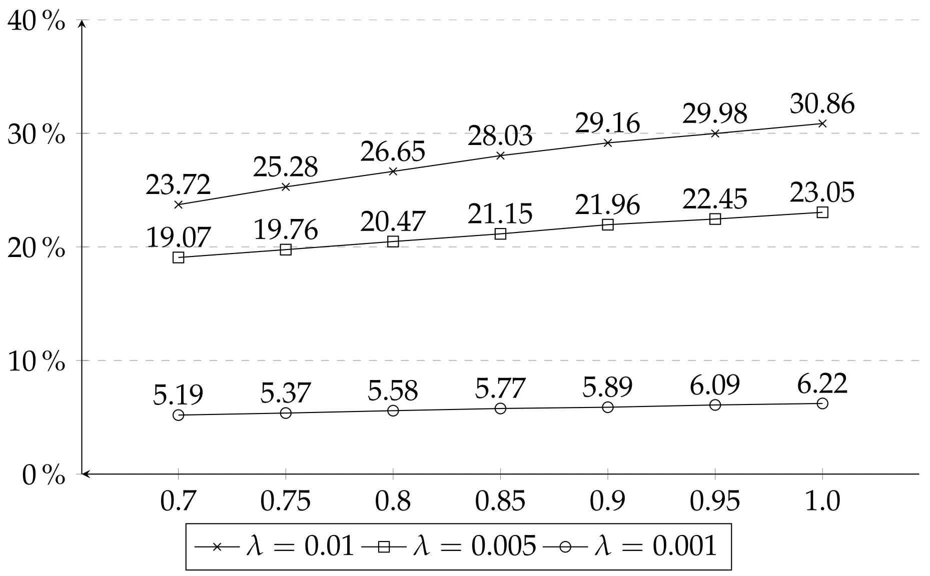


Figure 7.13: Idle listening reduction vs PDR for AllListen, Scenario 2

Network lifetime increase

Network lifetime increase is illustrated in figures 7.14 and 7.15. Lifetime increase is higher for both variants compared to Scenario 2.

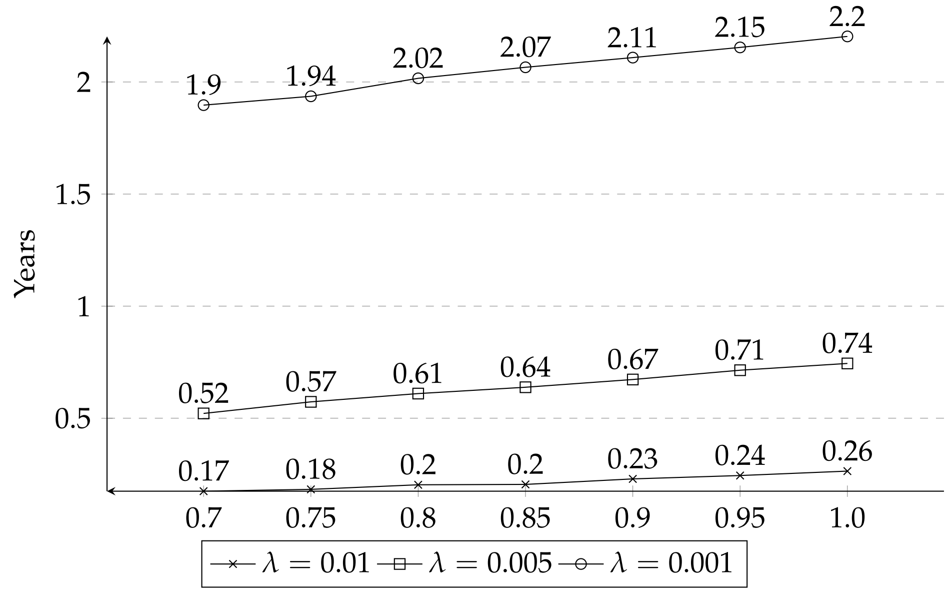


Figure 7.14: PDR vs increase in network lifetime for OneShot, Scenario 2

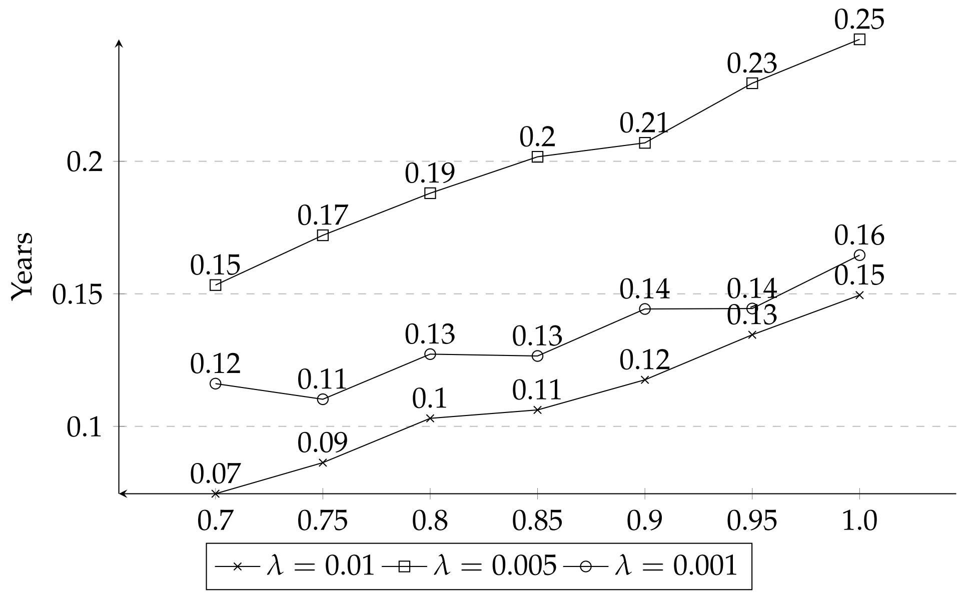


Figure 7.15: PDR vs increase in network lifetime for AllListen, Scenario 2

7.2.3 Analysis and discussion

The results of this scenario echo those of Scenario 1, but with lower values for latency and drops, and higher values of idle listening reduction and network lifetime increase. OneShot offers the most reduction overall at the cost of latency, and AllListen offers less reduction but without any noticeable latency penalty. OneShot still suffers from congestion for PDRs close to and lower than 0.8, but the amount of packets dropped has diminished, compared to Scenario 1. This is because OneShot now has 2 further cells available for TX/RX whenever a successful transmission of a packet with a pending bit is received, which gives queues a better chance at recovering from congestion.

Compared to the previous scenario, increase in network lifetime is higher for both TRA-variants, as seen in Figures 7.14 and 7.15. The gain is relatively high for OneShot, which now boasts a lifetime increase of 2.2 years compared to 1.27 in Scenario 1, for $\lambda = 0.001$. AllListen also demonstrates higher values, with the maximum value of 0.25 years compared to 0.13 years for $\lambda = 0.005$,.

It is clear that as more overprovisioned cells are added to the track, the potential for energy gain through reducing idle listening events becomes higher. In this case it is especially high as the max packet rates have not been changed. This means that when the PDR is high, the extra overprovisioning is unnecescary, as the extra cell ends up not used. However, for low PDRs, the extra overprovisioning does indeed help keep latency and packet drop lower than in Scenario 1.

7.3 Scenario 3 - Critical and Best-effort data

This section will detail the results of scenario 3, which intends to investigate the impact track cell reuse has on TRA. For this scenario, a network is considered consisting of a sink and two data-generating motes, where one mote is generating Critical data, while the other is generating best-effort data. Both critical and best-effort data use the same track to transport data from source to sink, but critical data has priority on the track. Critical data is created at a rate of $\lambda = 0.005$, and best-effort data is generated at a period of 1, 2 and 3 slotframes. The simulations were run with a PDR of 0.95 for each TRA variant along with a "None" baseline, in order to compare the results to the current state-of-the-art.

7.3.1 Results

Results are shown in Figure 7.16 for critical data, and Figure 7.17 for best-effort data, and lists the max and average latency and average packets dropped. Figure 7.18 shows the idle listening reduction observed for different data generation periods of best-effort packets. Figure 7.19 illustrates reported increase in network lifetime for both Oneshot and AllListen. Figure 7.20 shows the amount of drops for each TRA-variant per mote during data generation period 1.

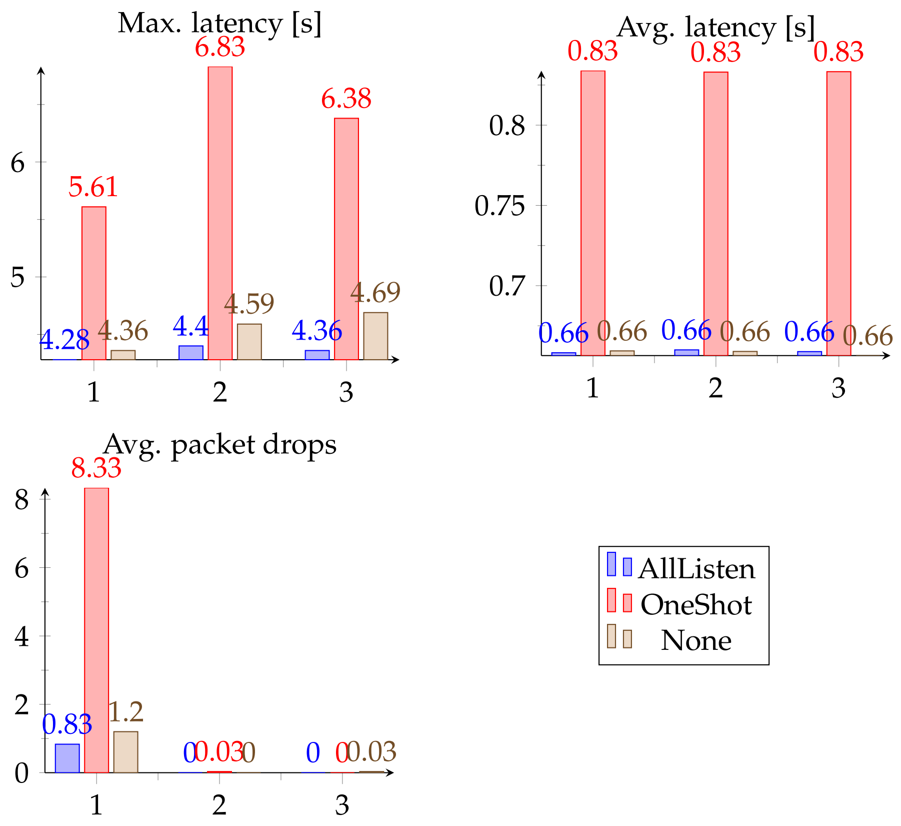


Figure 7.16: Critical data metrics vs traffic generation period for Scenario 3

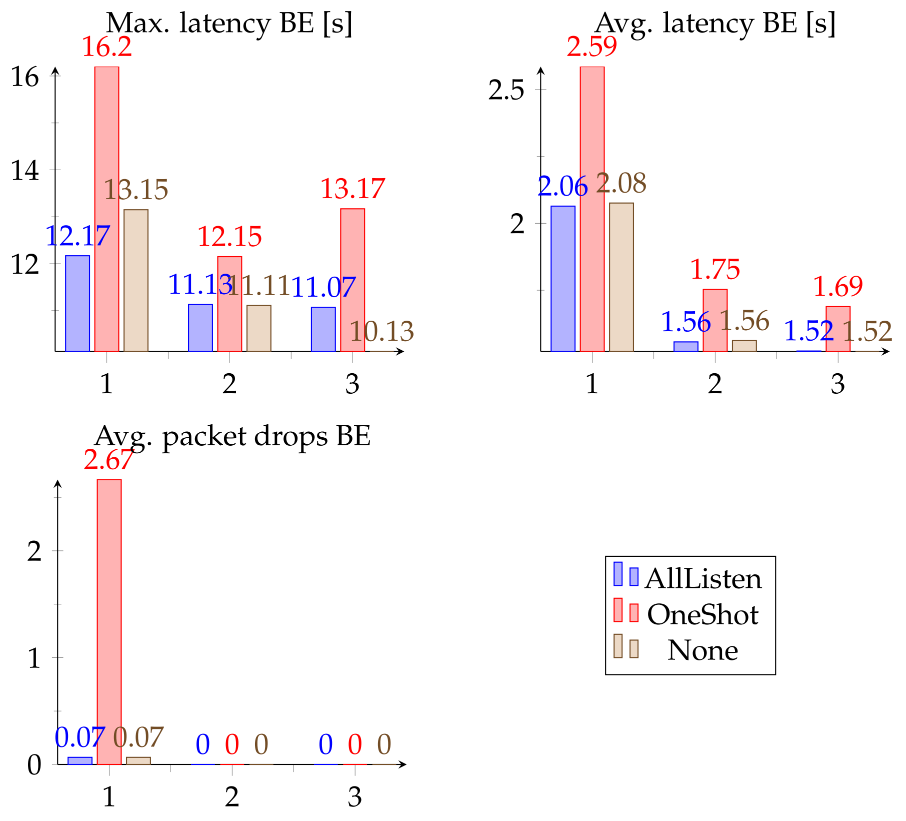


Figure 7.17: Best-effort data metrics vs traffic generation period for Scenario 3

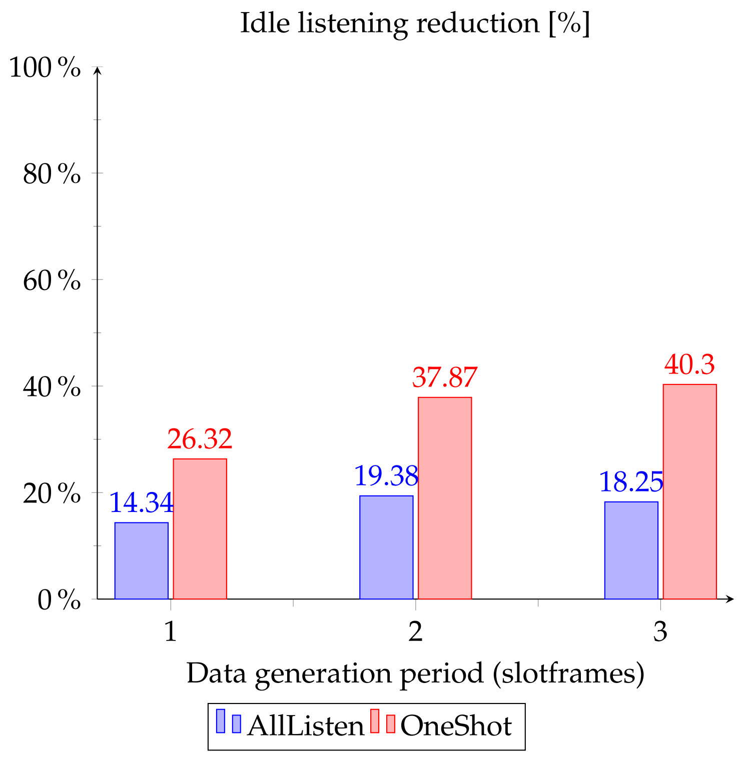


Figure 7.18: Idle listening reduction amount in percent for Scenario 3

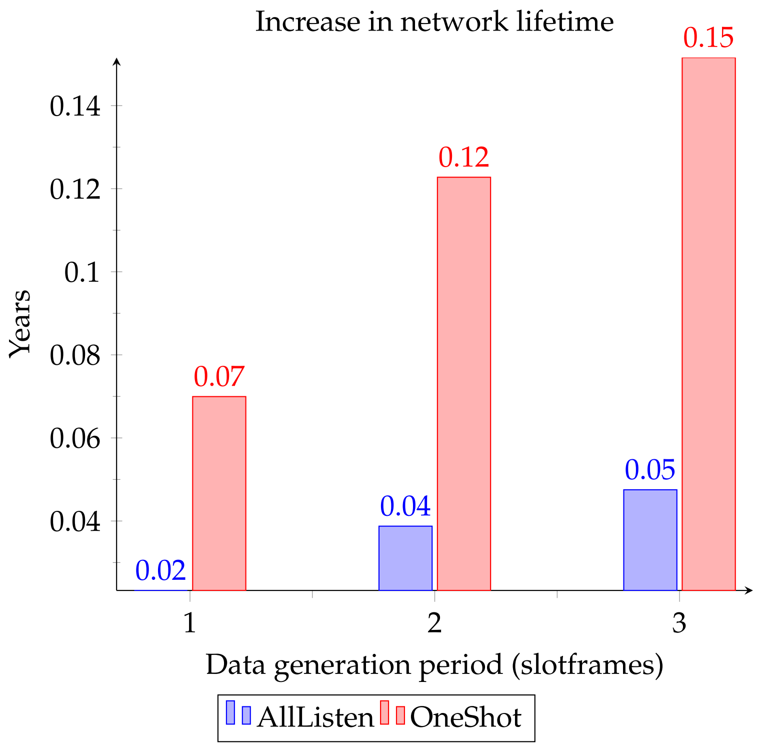


Figure 7.19: Increase in network lifetime, Scenario 3

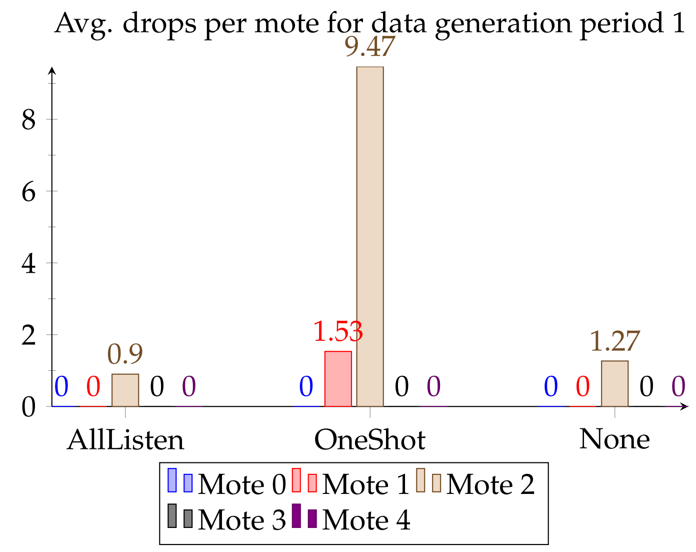


Figure 7.20: Average amount of drops per mote for data generation period 1, Scenario 3

7.3.2 Analysis and discussion

There is a clear difference between the two traffic types when it comes to latency. Critical data generally has lower latency than best effort, which is expected as critical data has priority. Thus, best-effort shows an increase in average latency as periods become shorter. Max latency for critical traffic spans 4.28s-4.69s for AllListen and None, while OneShot shows values ranging from 5.61s-6.83s. Average latency is constant at 0.66s for AllListen and None, but OneShot shows a higher value of 0.83s. OneShot also has the highest max and average latency for best-effort traffic.

As seen in Figure 7.18, OneShot offers the highest idle listening reduction for all periods of data generaration, with a value of 40.3% at period 3. AllListen shows the highest idle listening reduction at period 2 at 19.38%. Both have the lowest amount of reduction at period 1, with 14.34% and 26.32% for AllListen and OneShot respectively.

Network lifetime increase is displayed in Figure 7.19. The amount of increase is lower for both variants compared to the previous two scenarios. In the 6TiSCH simulator, the network lifetime metric is equal to the lowest expected lifetime observed among the motes in the network, which in this case would be mote 2. Mote 2 consumes a lot more energy compared to the other motes, as it must relay the traffic of mote 3 and 4. In addition, mote 2 features 2 extra cells from mote 4 that are not part of the track and cannot be disabled, resulting in excess idle listening.

When the traffic generation period is 1, drops occur for both packet types. As the PDR does not change in the scenario, the increase in packet drops are likely to stem from congestion. As presented in Figure 7.20, packet drops occur most often on mote 2. This would indicate that, for data generation period 1, mote 2 has trouble relaying traffic from mote 4 and 3 at a sufficient rate to avoid congestion. The issue is mostly apparent for OneShot, which is known from the previous scenarios to have trouble with congestion. In any way, the issue of congestion could be mitigated by increasing the number of overprovisioned cells in the track.

There is a higher amount of critical packets dropped compared to best effort, which warrants some explanation. The issue stems from the way cells are allocated in the scenario, and is illustrated in Figure 7.21. In short, best-effort traffic arrives at mote 2 after transmission opportunities to mote 1 have passed, meaning there is usually space cleared in the queue to accomodate a new packet. As critical data arrives in the cells prior to these transmission opportunities, the chance that the queue is full on reception is higher. When the queue is full, packets get dropped, and as the probability of a full queue is highest during the reception of critical data, more critical data is dropped in comparison to best-effort.

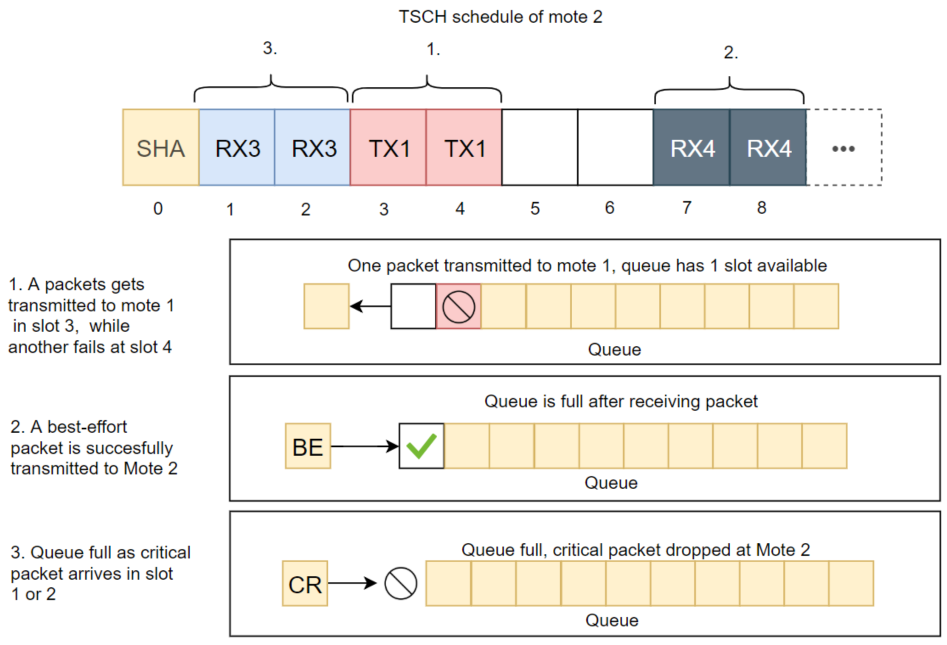

TSCH schedule of mote 2
3.
1.
2.
SHA
RX3
RX3
TX1
TX1
RX4
RX4
•••
0
1
2
3
4
5
6
7
8
1. A packets gets transmitted to mote 1 in slot 3, while another fails at slot 4
One packet transmitted to mote 1, queue has 1 slot available
Queue
2. A best-effort packet is succesfully transmitted to Mote 2
Queue is full after receiving packet
BE
Queue
3. Queue full as critical packet arrives in slot 1 or 2
Queue full, critical packet dropped at Mote 2
CR
Queue

Figure 7.21: Explanation of packet loss in scenario 3

Despite the lower increase in network lifetime shown in this scenario, TRA has the ability to offer a reduction in idle listening to networks with multiple traffic types. While OneShot does introduce caveats of increased latency and increased congestion loss, AllListen is able to provide a small increase in network lifetime at no cost to latency for both critical and best-effort packets.

Chapter 8

Conclusion

This thesis introduced and evaluated Track Resource Adaptation, which is a proposed mechanism intended to reduce idle listening events in 6TiSCH tracks featuring overprovisioning.

Two TRA variants, OneShot and AllListen, were introduced, and each variant was evaluated in three different simulation scenarios using the 6TiSCH simulator. The three scenarios represented different combinations of traffic rate and wireless link quality over varying amounts of overprovisioning, along with an evaluation of TRA in a network featuring different traffic classes. Metrics such as latency, packet loss, idle listening reduction and network lifetime increase were considered important to determine the benefits and drawbacks of TRA for each scenario.

The results indicate that OneShot is able to provide a high amount of idle listening reduction, and an increase in network lifetime of up to two years in some cases. However, this energy gain comes at the cost of higher latency and susceptibility to congestion loss when wireless links become more unreliable. AllListen induced minimal penalties to latency and packet loss, but provided less reduction in idle listening and a lower increase of network lifetime. For both variants, the potential energy gain becomes higher as the number of overprovisioned cells increase.

The applicability of the two TRA variants was discussed. AllListen is deemed suitable for a wider range of applications that could benefit from a small increase in network lifetime at little to no performance cost, while OneShot might be applicable in networks with good wireless links, and where higher overall latency is an acceptable tradeoff for a greater increase in network lifetime.

Chapter 9

Further work

As the scenarios defined in the thesis evaluate TRA in networks with a constant rate of packet generation, it would be interesting to also investigate how TRA responds to networks featuring mainly random bursts of traffic. For example, the thesis mentions that OneShot could offer a significant amount of idle listening reduction in such a scenario. Hence, further simulations representing scenarios with burst of traffic should be considered.

It would also be interesting to explore a different signaling mechanism. Rather than using the pending bit to toggle cells, a similar method to the one proposed by Fafoutis et. al. in [8] could be utilized, where a specific amount of cells are activated based on a integer value included in the header of a received packet. This could be used to only activate an (estimated) specific amount of cells needed rather than all, as is the current method.

Bibliography

[1] Diego Dujovne et al. '6TiSCH: deterministic IP-enabled industrial internet (of things)'. In: *IEEE Communications Magazine* 52.12 (2014), pp. 36–41.

[2] Jianping Song et al. 'WirelessHART: Applying wireless technology in real-time industrial process control'. In: *2008 IEEE Real-Time and Embedded Technology and Applications Symposium*. IEEE. 2008, pp. 377–386.

[3] *ISA100, Wireless Systems for Automation*. https://www.isa.org/isa100/. Accessed: 2019-01-01.

[4] 'IEEE Standard for Low-Rate Wireless Networks'. In: *IEEE Std 802.15.4-2015 (Revision of IEEE Std 802.15.4-2011)* (Apr. 2016), pp. 1–709. DOI: 10.1109/IEEESTD.2016.7460875.

[5] *IPv6 over the TSCH mode of IEEE 802.15.4e (6tisch)*. https://datatracker.ietf.org/wg/6tisch/about/. Accessed: 2019-01-01.

[6] *An Architecture for IPv6 over the TSCH mode of IEEE 802.15.4*. https://tools.ietf.org/html/draft-ietf-6tisch-architecture-20. Accessed: 2019-01-01.

[7] *Deterministic Networking Architecture*. https://tools.ietf.org/html/draft-ietf-detnet-architecture-12. Accessed: 2019-01-01.

[8] Xenofon Fafoutis et al. 'Adaptive static scheduling in IEEE 802.15. 4 TSCH networks'. In: *2018 IEEE 4th World Forum on Internet of Things (WF-IoT)*. IEEE. 2018, pp. 263–268.

[9] Yichao Jin et al. 'A centralized scheduling algorithm for IEEE 802.15. 4e TSCH based industrial low power wireless networks'. In: *2016 IEEE Wireless Communications and Networking Conference*. IEEE. 2016, pp. 1–6.

[10] Martin Wollschlaeger, Thilo Sauter and Juergen Jasperneite. 'The Future of Industrial Communication: Automation Networks in the Era of the Internet of Things and Industry 4.0'. In: *IEEE Industrial Electronics Magazine* 11.1 (2017), pp. 17–27. ISSN: 1932-4529. DOI: 10.1109/mie.2017.2649104.

[11] Hugh Boyes et al. 'The industrial internet of things (IIoT): An analysis framework'. In: *Computers in Industry* 101 (2018), pp. 1–12.

[12] *Definition: Industrial internet of things*. https://internetofthingsagenda.techtarget.com/definition/Industrial-Internet-of-Things-IIoT. Accessed: 2019-01-01.

[13] *Industrial IoT: How Connected Things are Changing Manufacturing*. https://www.ge.com/digital/blog/industrial-iot-how-connected-things-are-changing-manufacturing. Accessed: 2019-01-01.

[14] Maria Rita Palattella et al. 'On optimal scheduling in duty-cycled industrial IoT applications using IEEE802. 15.4 e TSCH'. In: *IEEE Sensors Journal* 13.10 (2013), pp. 3655–3666.

[15] Thomas Watteyne et al. 'Teaching communication technologies and standards for the industrial IoT? Use 6TiSCH!' In: *IEEE Communications Magazine* 55.5 (2017), pp. 132–137.

[16] Ian F Akyildiz et al. 'A survey on sensor networks'. In: *IEEE Communications magazine* 40.8 (2002), pp. 102–114.

[17] Priyanka Rawat et al. 'Wireless Sensor Networks: recent developments and potential synergies'. In: *Journal ofSuperComputing, www. researchgate. net/publication/25* 8165429 ().

[18] Vehbi C Gungor and Gerhard P Hancke. 'Industrial wireless sensor networks: Challenges, design principles, and technical approaches'. In: *IEEE Transactions on industrial electronics* 56.10 (2009), pp. 4258–4265.

[19] Rodrigo Teles Hermeto, Antoine Gallais and Fabrice Theoleyre. 'Scheduling for IEEE802. 15.4-TSCH and slow channel hopping MAC in low power industrial wireless networks: A survey'. In: *Computer Communications* 114 (2017), pp. 84–105.

[20] *Industrial wireless networks - comparing the standards*. https://www.processonline.com.au/content/wireless/article/industrial-wireless-networks-mdash-comparing-the-standards-part-2-202802088. Accessed: 2019-01-01.

[21] Domenico De Guglielmo, Simone Brienza and Giuseppe Anastasi. 'IEEE 802.15.4e: A survey'. In: *Computer Communications* 88 (2016), pp. 1–24. ISSN: 0140-3664. DOI: https://doi.org/10.1016/j.comcom.2016.05.004. URL: http://www.sciencedirect.com/science/article/pii/S0140366416301980.

[22] 'IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer'. In: *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)* (Apr. 2012), pp. 1–225. DOI: 10.1109/IEEESTD.2012.6185525.

[23] Domenico De Guglielmo, Giuseppe Anastasi and Alessio Seghetti. 'From IEEE 802.15.4 to IEEE 802.15.4e: A step towards the Internet of Things'. In: *Advances in Intelligent Systems and Computing* 260 (Mar. 2014), pp. 135–152. DOI: 10.1007/978-3-319-03992-3_10.

[24] Klaus Wehrle, Mesut Günes and James Gross. *Modeling and tools for network simulation*. Springer Science & Business Media, 2010.

[25] Esteban Municio et al. 'Simulating 6TiSCH networks'. In: *Transactions on Emerging Telecommunications Technologies* 30.3 (2019), e3494.

[26] *OMNeT++ Discrete Event Simulator*. https://omnetpp.org/. Accessed: 2019-01-01.

[27] *ns3 | a discrete event simulator for Internet systems*. https://www.nsnam.org/. Accessed: 2019-01-01.

[28] *GitHub - IRC-SPHERE/tsch-simulator*. https://github.com/IRC-SPHERE/tsch-simulator. Accessed: 2019-01-01.

[29] Xavier Vilajosana et al. 'A realistic energy consumption model for TSCH networks'. In: *IEEE Sensors Journal* 14.2 (2014), pp. 482–489.

[30] Xavier Vilajosana et al. 'OpenMote: Open-source prototyping platform for the industrial IoT'. In: *International Conference on Ad Hoc Networks*. Springer. 2015, pp. 211–222.

[31] Maria Rita Palattella et al. 'On-the-fly bandwidth reservation for 6TiSCH wireless industrial networks'. In: *IEEE Sensors Journal* 16.2 (2016), pp. 550–560.

[32] Seema Kharb and Anita Singhrova. 'Slot-frame Length Optimization using Hill Climbing for Energy Efficient TSCH Network'. In: *Procedia computer science* 132 (2018), pp. 541–550.

[33] Glenn Daneels et al. 'Accurate Energy Consumption Modeling of IEEE 802.15. 4e TSCH Using Dual-Band OpenMote Hardware'. In: *Sensors* 18.2 (2018), p. 437.

[34] Kanghoon Choi and Sang-Hwa Chung. 'Enhanced time-slotted channel hopping scheduling with quick setup time for industrial Internet of Things networks'. In: *International Journal of Distributed Sensor Networks* 13.6 (2017), p. 1550147717713629.

[35] *Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration*. https://tools.ietf.org/html/rfc8180. Accessed: 2019-01-01.

[36] Fabrice Theoleyre and Georgios Z. Papadopoulos. 'Experimental Validation of a Distributed Self-Configured 6TiSCH with Traffic Isolation in Low Power Lossy Networks'. In: *Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*. MSWiM '16. Malta, Malta: ACM, 2016, pp. 102–110. ISBN: 978-1-4503-4502-6. DOI: 10.1145/2988287.2989133. URL: http://doi.acm.org/10.1145/2988287.2989133.

Appendix A

Acronyms

6LoWPAN	IPv6 over Low-Power Wireless Personal Area Networks
6P	6top protocol
6TiSCH	IPv6 over the TSCH mode of IEEE 802.15.4
6top	6TiSCH Operation Sublaryer
ACK	Acknowledgment packet
ASN	Absolute slot number
BE	Best Effort
CCA	Clear Channel Assesment
CSMA-CA	Carrier Sense Multiple Access Collision Avoidance
CoAP	Constrained Application Protocol
DAO	Destination Advertisement Object
DIO	Destination Information Object
EB	Enhanced Beacon
ETX	Estimated Number of Transmissions
FDMA	Frequency Division Multiple Access
FES/FEL	Future Event Set or Future Event List
FFD	Full-Function Device
G-MPLS	Generalized Multi-Protocol Label Switching
GTS	Guaranteed timeslot
IACS	Industrial Automation and Control Systems
IEEE	Institute of Electrical and Electronics Engineers

IETF	Internet Engineering Task Force
IIoT	Industrial Internet-of-Things
IPv6	Internet Protocol version 6
IP	Internet Protocol
IT	Information Technology
IWSN	Industrial Wireless Sensor Network
IoT	Internet-of-Things
KPI	Key Performance Indicator
KPI	Key Performance Indicator
LLN	Low-power and lossy networks
LR-WPAN	Low-Rate Wireless Personal Area Network
MAC	Medium Access Control
MAC	Medium Access Control
MPDU	MAC Protocol Data Unit
MSF	Mininmal Scheduling Function
NME	Network Management Entity
OT	Operational Technology
PAN	Personal Area Network
PCE	Path Computation Entity
PDR	Packet Delivery Ratio
PHY	Physical, layer 1 of the OSI-model
PPDU	PHY Protocol Data Unit
PREOF	Packet Replication and Ordering Functions
QoS	Quality - of - Service
RFD	Reduced-Function Device
RF	Radio Frequency
RPL	Routing Protocol for Low-Power and Lossy Networks
RSSI	Received Signal Strength Indicator
RX	Reception

SF	Scheduling function
TDMA	Time Division Multiple Access
TRA	Track Resource Adaptation
TSCH	Time Slotted Channel Hopping
TX	Transmission
UUID	Universally Unique Identifier
WG	Working Group
WSN	Wireless Sensor Network

Appendix B

Python code

The modified 6TiSCH simulator used in this thesis, along with various helper scripts is uploaded to GitHub.com

https://github.com/mutg/track_resource_adaptation

Appendix C

Additional results

C.1 Scenario 1

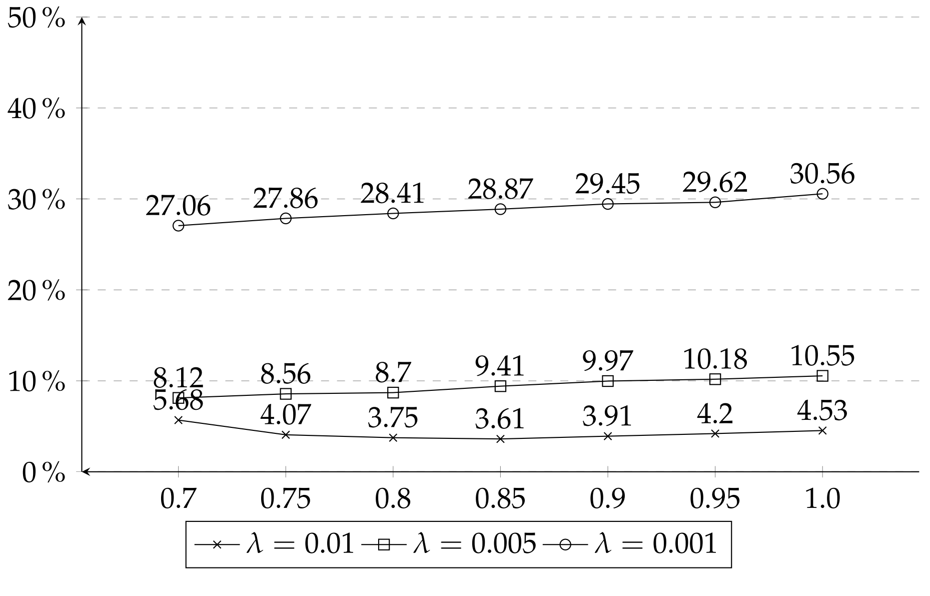


Figure C.1: PDR vs percentage reduction in charge consumed for OneShot, Scenario 1

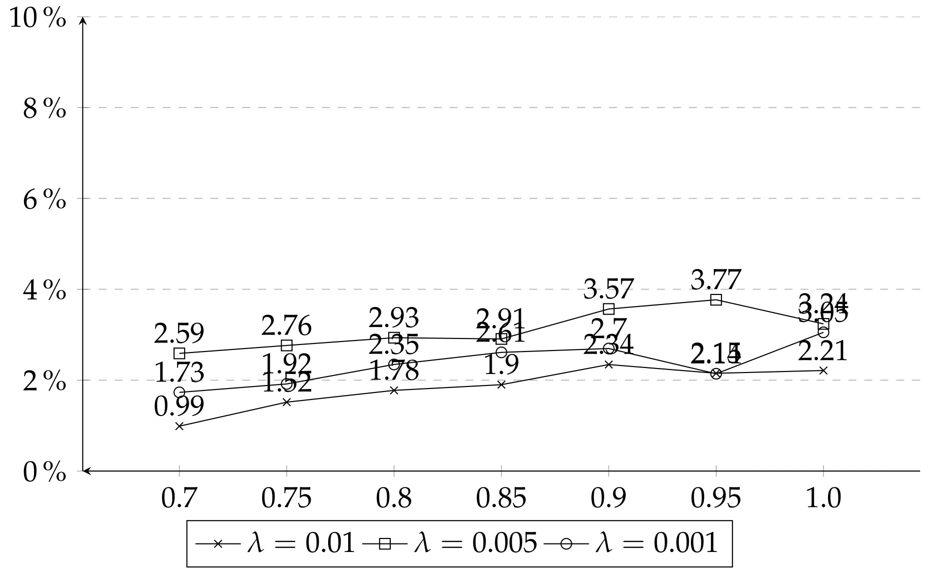


Figure C.2: PDR vs percentage reduction in charge consumed for AllListen, Scenario 1

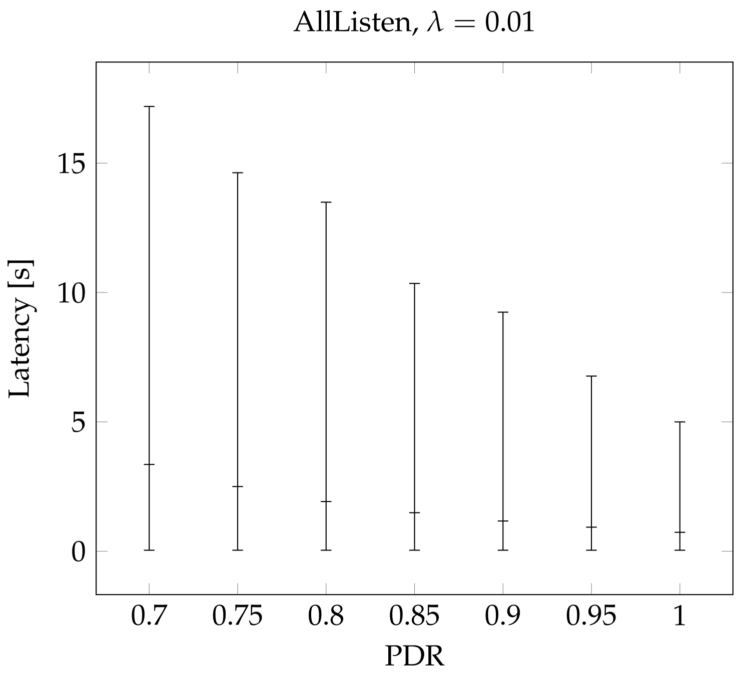


Figure C.3: Min max and average latency, AllListen, $\lambda = 0.01$, Scenario 1

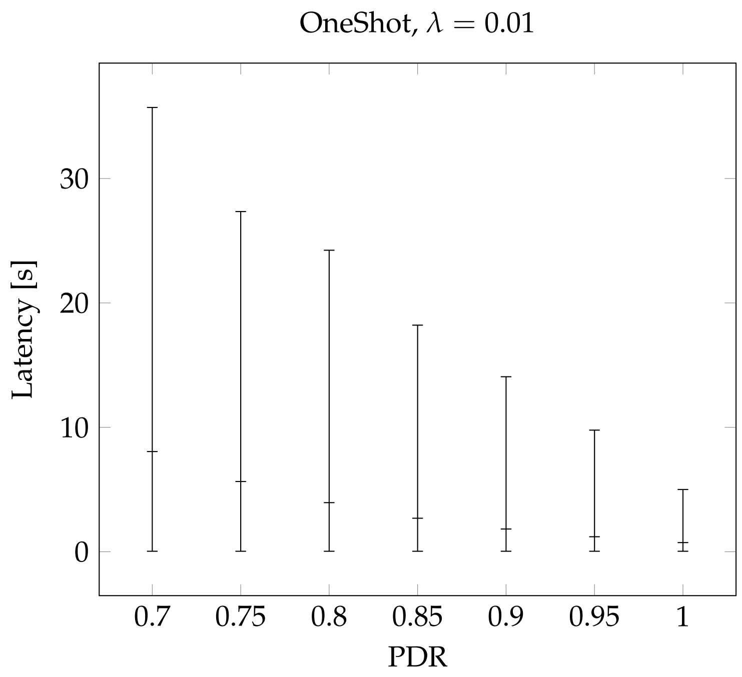
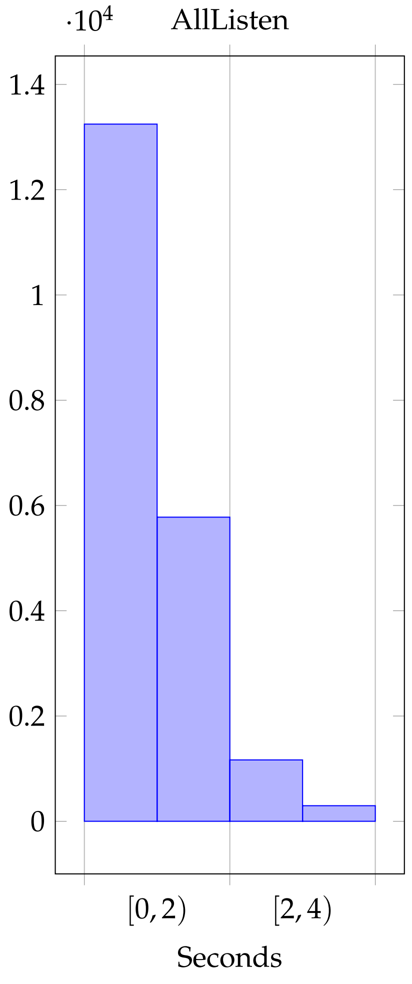
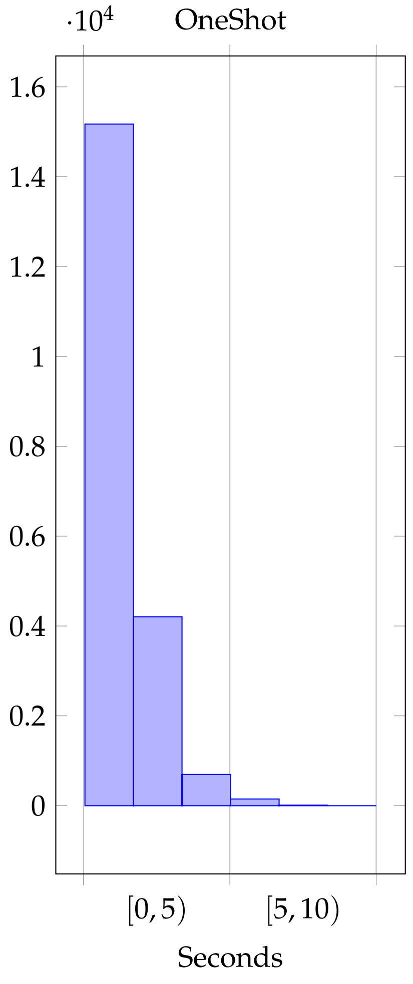


Figure C.4: Min max and average latency, OneShot, $\lambda = 0.01$, Scenario 1



(a)



(b)

Figure C.5: Latency distrubution for PDR=0.95, $\lambda = 0.01$, Scenario 1

C.2 Scenario 2

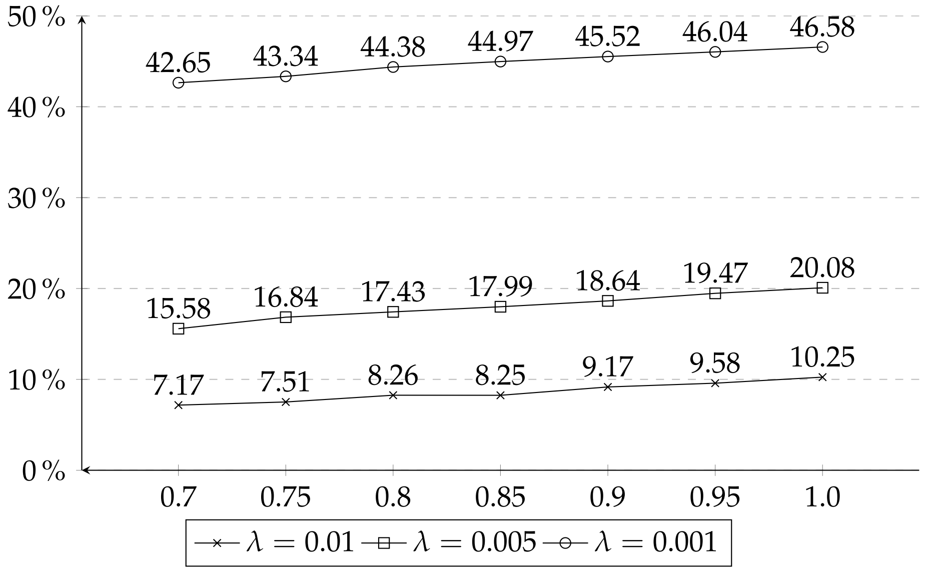


Figure C.6: PDR vs percentage reduction in charge consumed for OneShot, Scenario 2

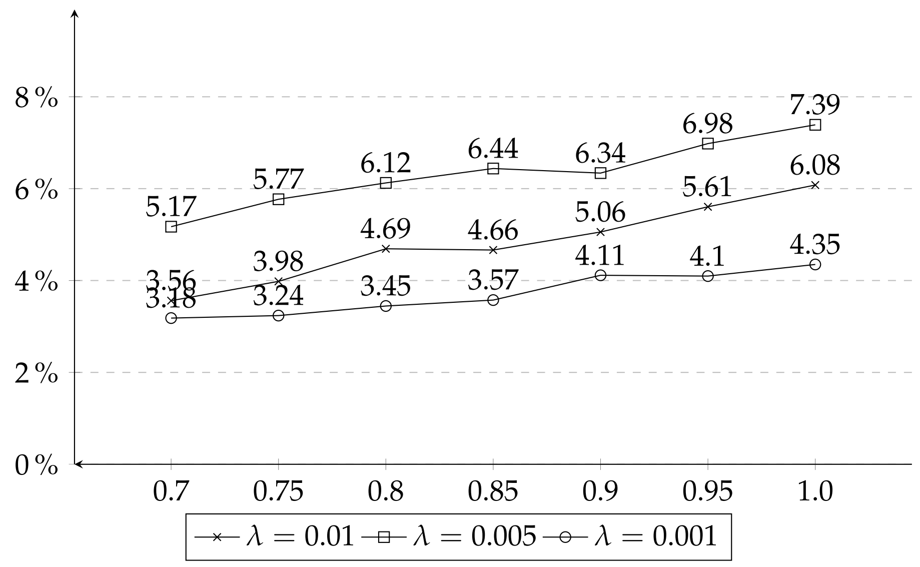


Figure C.7: PDR vs percentage reduction in charge consumed for AllListen, Scenario 2

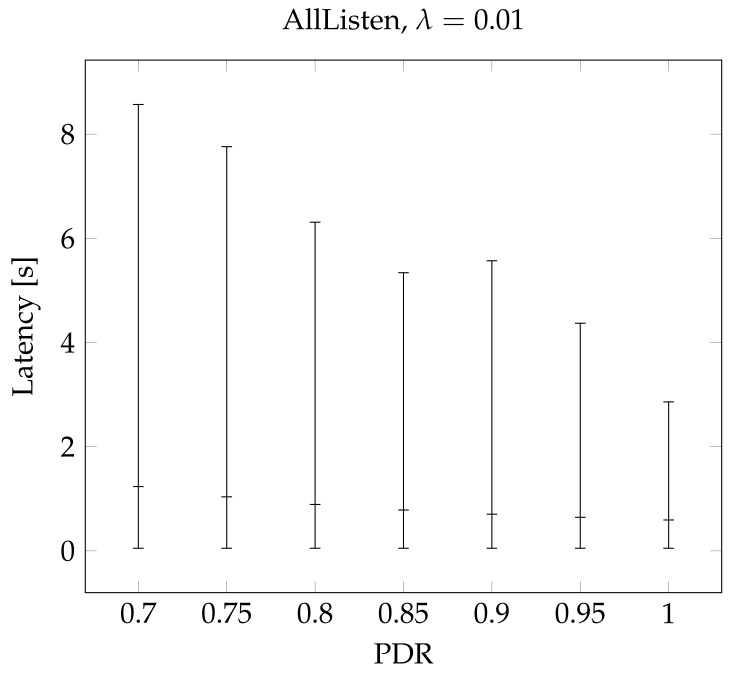


Figure C.8: Min max and average latency, AllListen, $\lambda = 0.01$, Scenario 2

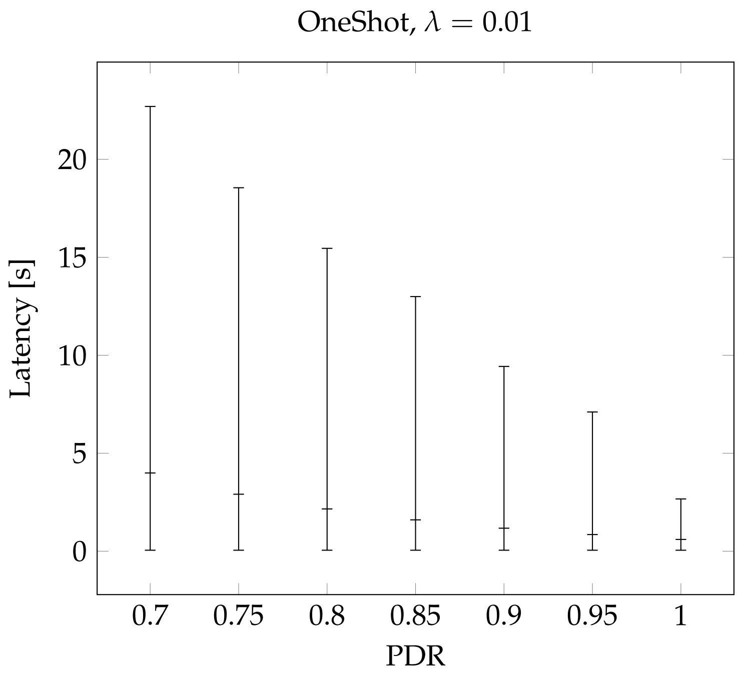
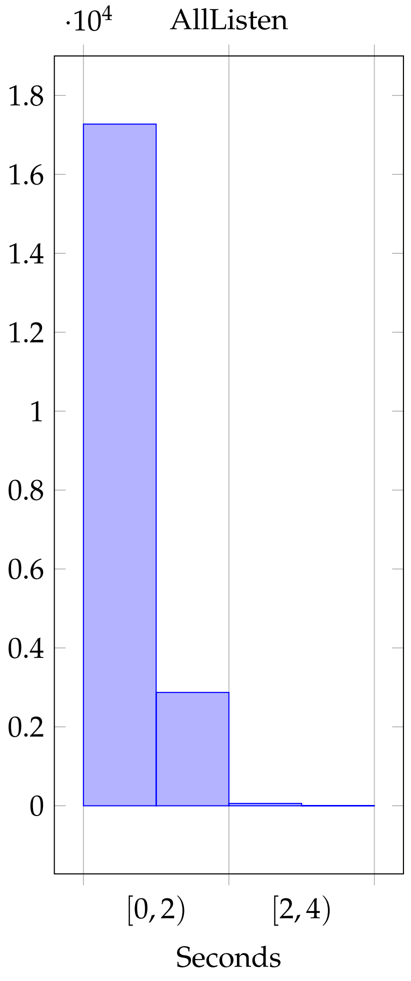
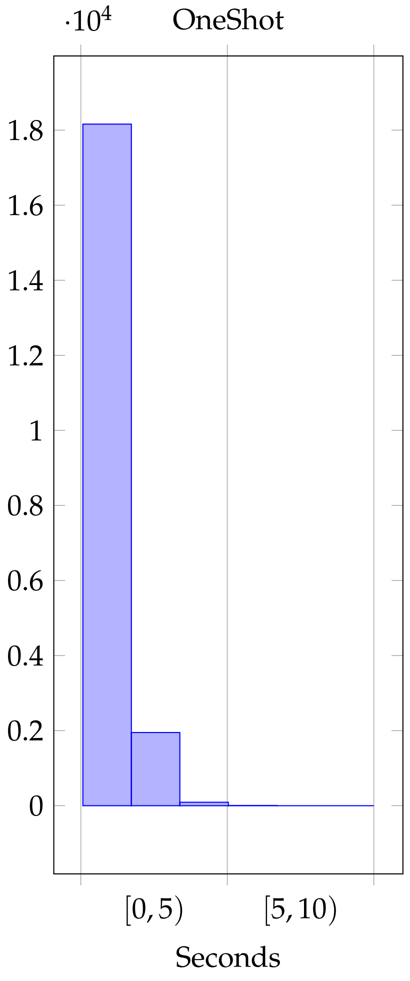


Figure C.9: Min max and average latency, OneShot, $\lambda = 0.01$, Scenario 2



(a)



(b)

Figure C.10: Latency distrubution for PDR=0.95, $\lambda = 0.01$, Scenario 2

C.3 Scenario 3

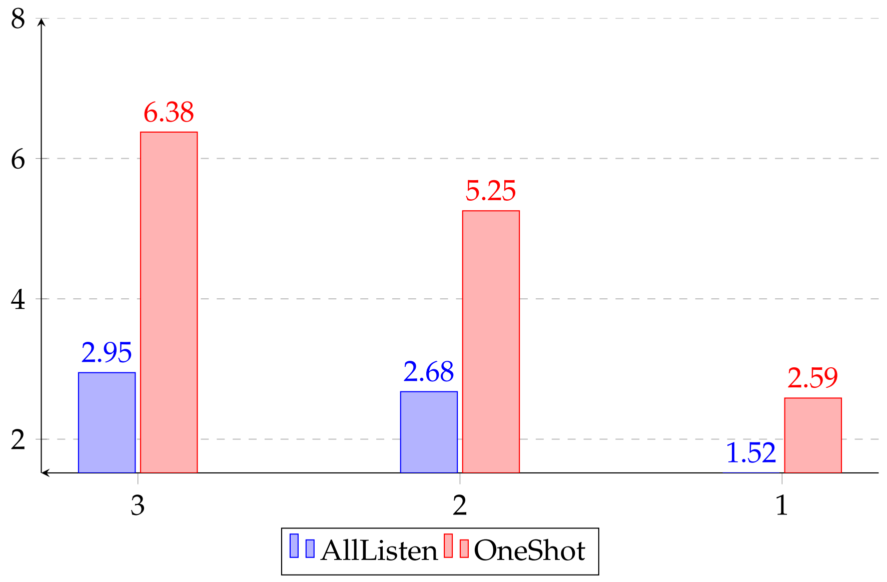


Figure C.11: PDR vs percentage reduction in charge consumed for AllListen and OneShot, Scenario 3

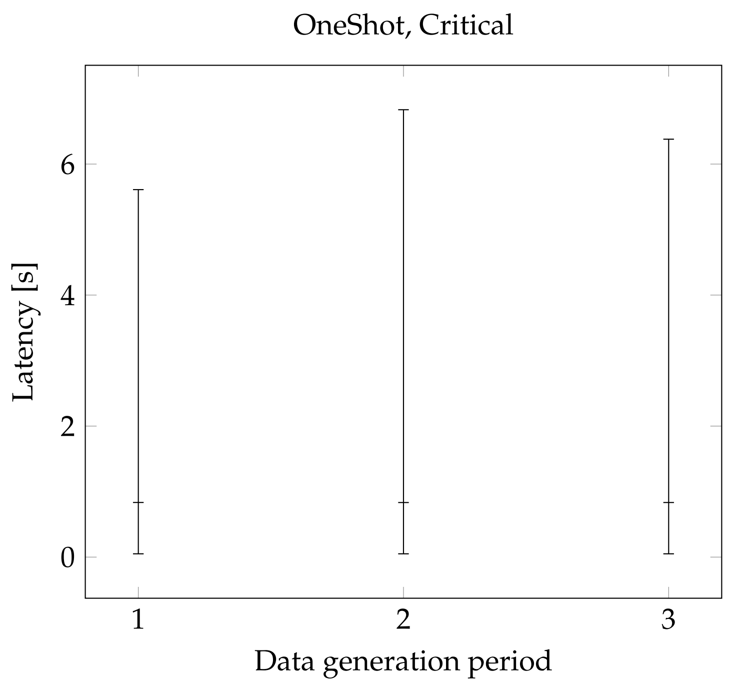


Figure C.12: Min max and average latency, OneShot, Critical data, Scenario 3

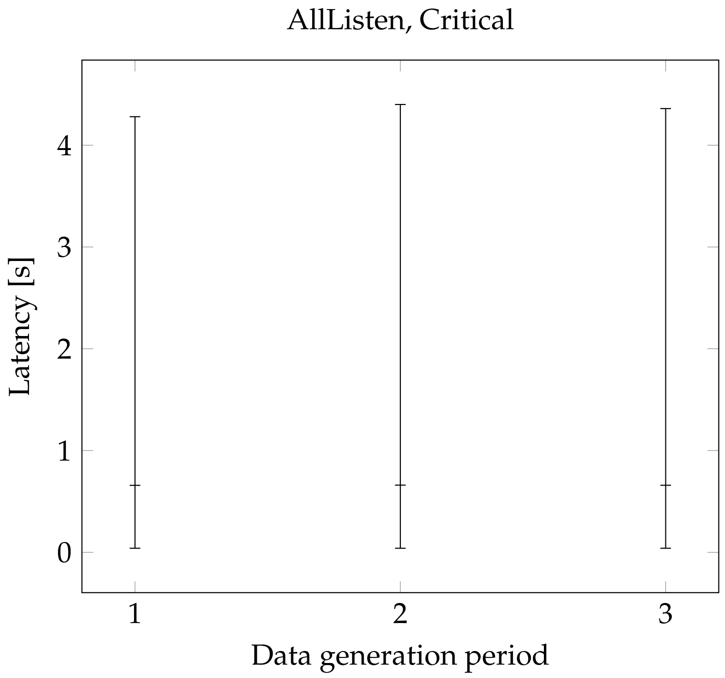
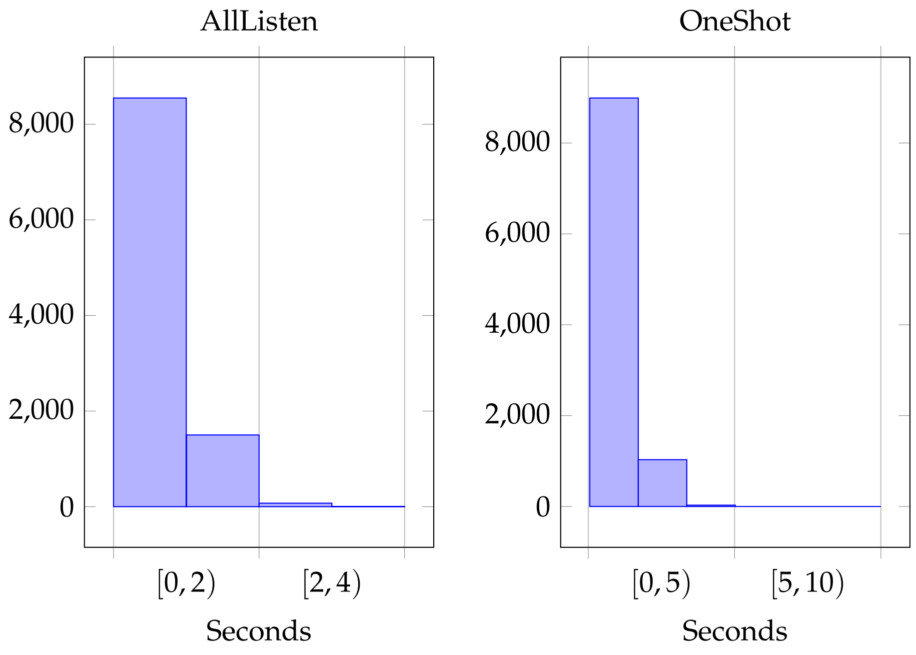
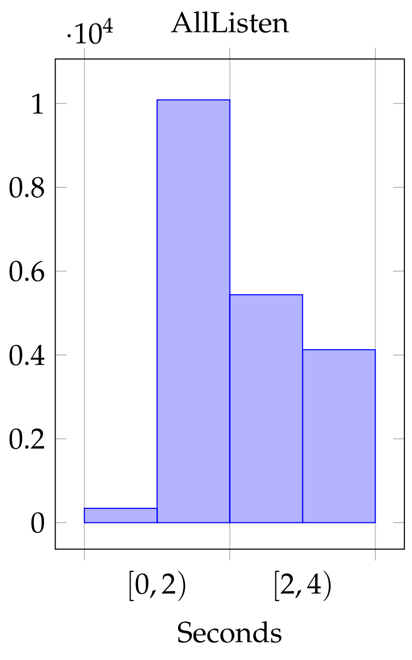


Figure C.13: Min max and average latency, AllListen, Scenario 3

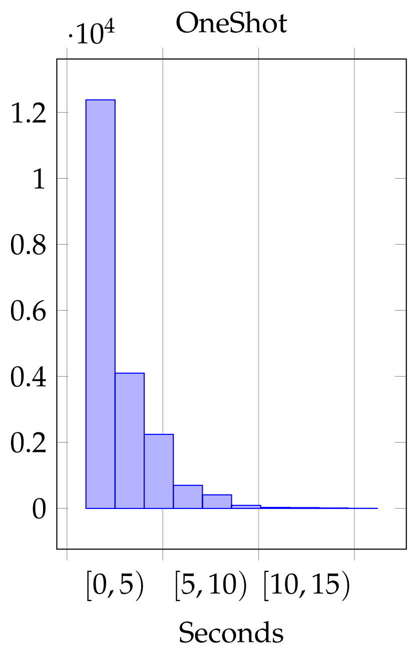


(a) **(b)**

Figure C.14: Latency distrubution for critical traffic at data generation period 1, Scenario 3



(a)



(b)

Figure C.15: Latency distrubution for best effort traffic at data generation period 1, Scenario 3