TUHH
Hamburg University of Technology

ComNets
Institute of Communication Networks

Bachelor Thesis

Evaluation of the Cell Allocation Mechanism in 6TiSCH Minimal Scheduling Function for Wireless Sensor Networks

Benjamin Lih-Hsiang Ko

25.03.2025

Evaluation of the Cell Allocation Mechanism in 6TiSCH Minimal Scheduling Function for Wireless Sensor Networks

Benjamin Lih-Hsiang Ko
Matriculation number 528180
Informatik Ingenieurwesen B.Sc.

Hamburg University of Technology
Institute of Communication Networks
First examiner: Prof. Dr.-Ing. Timm-Giel
Second examiner: Dr.-Ing. Koojana Kuladinithi
Supervisor: Yevhenii Shudrenko

Hamburg, 25.03.2025

Declaration of Originality

I hereby declare that the work in this thesis was composed and originated by myself and has not been submitted for another degree or diploma at any university or other institute of tertiary education.

I certify that all information sources and literature used are indicated in the text and a list of references is given in the bibliography.

Hamburg, 25.03.2025

Benjamin Lih-Hsiang Ko

Abstract

As the world becomes increasingly connected through smart technologies and the Internet of Things (IoT), Wireless Sensor Networks (WSNs) have emerged as a fundamental technology for enabling efficient environmental data collection and transmission. To meet the demands for scalability, self-management, and low power consumption, the IEEE 802.15.4 standard introduced the Time-Slotted Channel Hopping (TSCH) Medium Access Control (MAC) protocol, forming the basis of the IPv6 over the Time-Slotted Channel Hopping mode of IEEE 802.15.4e (6TiSCH) protocol stack. Within this stack, the Minimal Scheduling Function (MSF) plays a crucial role in managing cell allocation. This work investigates two cell allocation mechanisms under varying network interference and MSF parameters. An analytical model is proposed to predict cell allocation time, and its accuracy is validated through experimental results. The results show that the sensing cell allocation mechanism reduces cell overlaps, resulting in faster allocation times and eliminating the need for cell relocations. Additionally, network interference is observed to influence allocation time, regardless of the mechanism used.

Contents

1 Introduction

In a world where ubiquitous computing and the IoT are becoming increasingly more important, WSNs have risen to become a key technology making it possible to accurately digitize the environment and collect data into a centralized cloud. Although with Wi-Fi and Bluetooth there were already pre-existing technologies that support wireless communication, but their characteristics do not suit the needs of WSNs. WSNs need to be scalable, self-managing and low power since most of the devices will be battery powered, which both of the before mentioned protocols are not. That is why in 2009 the IEEE 802.15.4 standard [1] was published which defines the MAC sublayer and the physical layer of a Low Power and Lossy Network (LLN). This standard was later extended by several amendments such as IEEE 802.15.4e, which added the TSCH MAC protocol.

Based on the IEEE 802.15.4 standard and the TSCH MAC protocol a new protocol stack for wireless communication was developed by the IETF. This new protocol stack is called 6TiSCH which is a combination of IPv6 and TSCH. It enables IEEE 802.15.4 networks to use IPv6, which they are not natively capable of, since they are optimized for low power and low data whereas IPv6 is designed for handling medium to large amounts of data. Within this 6TiSCH protocol the Minimal Scheduling Function (MSF) plays a big part in regulating the protocols behaviour. Since the MSF is standartized by an RFC it has been studied in several papers already.

This work intends to add missing research, namely the evaluation of the cell allocation mechanism of MSF. Using a analytical model and experimental validation this work analyzes the relationship between various network scenarios and the time it takes for the network to allocate cells and stabilize again. This relationship is investigated for two variations of cell allocation mechanisms.

The following sections will first introduce the necessary background for the research then give an overview of the already existing work and related work in this field. After that the analytical model is introduced followed by the experimental validation and a discussion on the results.

2 Background

2.1 IEEE 802.15.4

The IEEE 802.15.4 standard [1] is a wireless network standard published in 2009 defining the MAC sublayer and physical layer of a LLN. It's purpose is to provide an alternative to existing wirless technologies such as Wifi or Bluetooth for low power, low cost, low complexity and low data rate wireless networks. It is designed for short range communication for tens of meters and can be used on different frequency bands with the publicly available 2.4 GHz one being the most commonly used. With 16 available channels the connection allows for a data rate of 250 kbits/s, which compared to the Mbits/s data rate of the Wifi standard is considered a low data rate connection.

The MAC sublayer of the IEEE 802.15.4 protocol defines several variants, such as the default Carrier-sense Multiple Access with Collision Avoidance (CSMA/CA) but also the later with the IEEE 802.15.4e amendment [2] added TSCH MAC protocol. Throught the years several protocol stacks have emerged, that build on this standard, namely Zigbee,WirelessHART and 6TiSCH.

2.1.1 Time-Slotted Channel Hopping (TSCH) MAC protocol

The Time-Slotted Channel Hopping (TSCH) MAC protocol [2] was introduced as an alternative to the default CSMA/CA MAC protocol. Through it's design it enables the wireless connection to be deterministic [3] and therefore more predictable and reliable. This is achieved through a combination of Time Division Multiplexing (TDM) and Frequency Division Multiplexing (FDM) creating a matrix of so called cells. Each cell is identified by a unique set of slotOffset and channelOffset. In the time domain the TSCH schedule is divided into repeating slotframes, which themselves are again subdivided into timeslots. For the FDM aspect of the protocol as before mentioned each cell has an assinged channelOffset. The number of these correspond to the available frequency channels. Since TSCH include channel hopping the actual physical channel for each channelOffset changes with each passing slotframe. The corresponding channel for each channelOffset is calculated as follows:

$$channel = (ASN + chOf) \mod n_{ch} \tag{2.1}$$

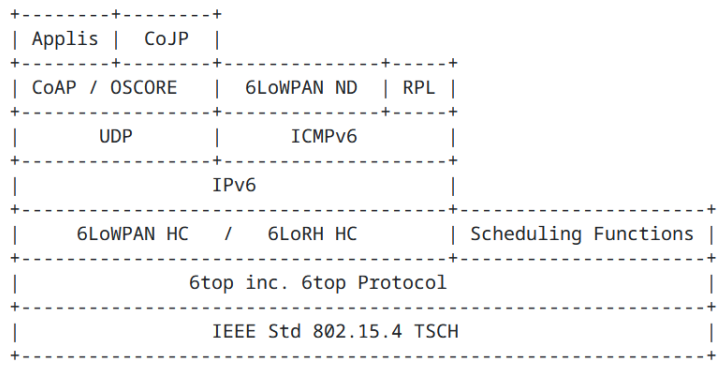

Figure 2.1: 6TiSCH protocol stack defined by RFC9030. [4]

with ASN being the Absolute Slot Number (ASN), that is how many timeslots have passed since the start of the network, chOf being channelOffset of the cell and n_{ch} being the total number of channels available. The channel hopping mechanism mitigates interference on a specific frequency therefore making the network more robust. It also lessens the effects of multipath fading, making the network more reliable in changing environments.

Cells can be allocated as dedicated or shared cells. Dedicated cells are reserved for unicast transmissions of messages only between two cells, which guarantees a contention free communication. On the other hand shared cells are allocated for all nodes to communicated on. In these cells the TSCH CSMA/CA protocol is used to avoid collisions. It is in these cells that Enhanced Beacons (EBs) are broadcast, that carry network-specific data essential for new nodes to join the network and for the overall synchronization of the nodes.

2.2 IPv6 over the TSCH mode of IEEE 802.15.4e (6TiSCH)

Since IEEE 802.15.4 is only a protocol defining the MAC sublayer and the physical layer, a network that intends to connect to the wider internet needs to be able to have IP connectivity. This is what IPv6 over the Time-Slotted Channel Hopping mode of IEEE 802.15.4e (6TiSCH) [4] aims to achieve. It is a protocol stack that provides IEEE 802.15.4 standard using the TSCH MAC protocol with mechanisms to manage its schedule centralized or decentralized and IP connectivity. By defining a sublayer for schedule management above the IEEE 802.15.4 standard and integrating various technologies on top of that 6TiSCH makes IPv6 viable for low power and low data rate networks. The protocol stack enabling this addition of IPv6 can be seen in Figure 2.1.

The following sections will delve deeper into the most relevant components of the protocol stack defined by 6TiSCH.

2.2.1 6LoWPAN Adaptation Layer

Since IPv6 is built on the assumption that the Maximum Transmission Unit (MTU) is at least 1280 bytes or above, there is the need to make it compatible with the actual MTU of the IEEE 802.15.4 protocol, which is only 127 bytes. In addition the normal IPv6 header size of 40 bytes is too big for such a small data rate, since it alone would already take up around 30% of the available space.

In order to make IPv6 compatible with IEEE 802.15.4 6TiSCH makes use of IPv6 over Low-Power Wireless Personal Area Networks (6LoWPAN), which has two main mechanisms for handling the above mentioned problems. First 6LoWPAN applies Header Compression (HC) [5] to the IPv6 header reducing it from 40 bytes to just a few bytes, by for instance omitting unnecessary or redundant parts of the header. Secondly in case a package is bigger than the MTU of the link layer 6LoWPAN handles the fragmentation and reassembly [6] of data packets. By applying these techniques 6LoWPAN makes IPv6, which is primarily designed for comparatively high data rate networks, viable for the IEEE 802.15.4 protocol.

2.2.2 RPL

The Routing Protocol for Low-Power and Lossy Networks (RPL) [7] is a crucial part of the 6TiSCH stack to ensure reliable and efficient routing in a multi hop and decentralized network. By creating and maintaing a Destination-Oriented Directed Acyclic Graph (DODAG) it enables the network to route packets intelligently and efficiently. The RPL root is the central node which coordinates the network and also serves as gateway to external networks. The other nodes in the network communicate upwards to the root via a so called preferred parent, which is determined by listening to DODAG Information Objects (DIOs) and according to e.g. link quality pick the preferred parent. In the joining process of a new node RPL also plays a vital role. One of the first actions the new node does is to listen to DIOs and then choose it's preferred parent. Having done that the node then is able to send and recieve IPv6 packets. As can be seen in Figure 2.2 the nodes are classified into ranks depending on their distance to the sink. Not ony do parent nodes send DIOs downstream, but Destination Advertisement Object (DAO) messages are used by the child nodes to advertivse routes from the leave or intermediate node to the root. These messages inform the root about the structure of the DODAG and help to keep it updated.

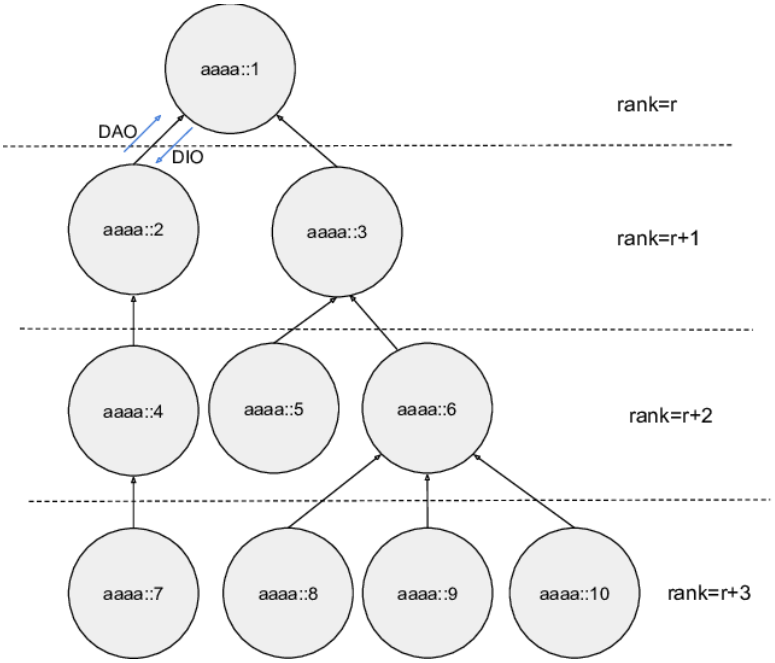

Figure 2.2: RPL DODAG showing rang and messages. [8]

2.2.3 6TiSCH Operation Sublayer (6top) and 6top Protocol (6P)

The so called 6top [9] is the connecting layer between the IEEE 802.15.4e MAC protocol and the 6LoWPAN implementation of IPv6. It is the main contribution of the 6TiSCH protocol and it provides the tools to manage the TSCH schedule in a centralized or decentralized manner. It is comprised of two major components firstly the 6top Protocol (6P) [9], which is a protocol for negotiating cells between nodes and synchronizing the schedules and secondly the Scheduling Function (SF) which runs on each node and is responsible for deciding when to add, delete or relocate cells. Cells are categorized into hard and soft cells. Hard cells cannot be changed by 6top whereas soft cells can be managed by 6top.

The 6P provides several message types for two nodes to negotiate, manage and synchronize their TSCH schedule. The available 6P commands and their functions are listed in Table 2.1. Using 6P commands the so called 6P transactions can be made between two nodes for example to add cells to the schedule. 6P defines two variants of the 6P add transaction, which are the two way and the three way handshake. For this thesis only the two way handshake will be relevance hence we will only be considering this variant of transaction. For the 6P add transaction first an add request is sent by node A containing a list of candidate cells and the amount of cells that the node wants to schedule for communication. Upon recieving the request node B then evaluates which cells are preferred for communication and then sends a response back to node A containing

Table 2.1: 6P commands as defined by RFC8480 [9] taken from [10]

Command	Description
ADD	Allocate new cells with a peer
DELETE	Deallocate cells scheduled with a peer
RELOCATE	Move cells scheduled with a peer
CLEAR	Deallocate all the cells scheduled with a peer
COUNT	Return the number of cells scheduled with a peer
LIST	Return the list of cells scheduled with a peer
SIGNAL	Communicate arbitrary data with a peer

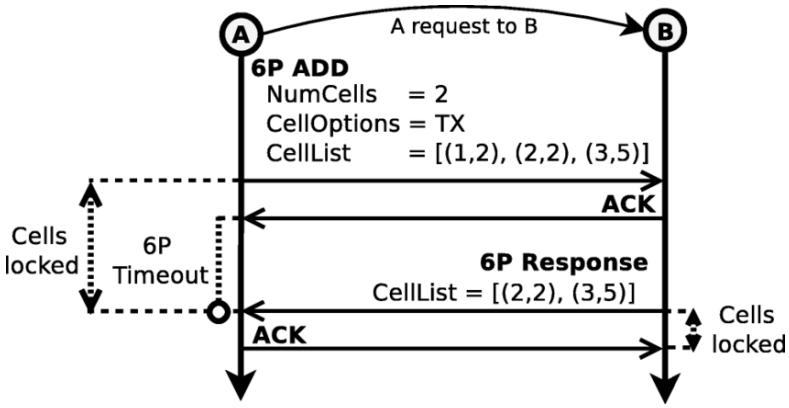

Figure 2.3: Process of two way 6P ADD transaction. [11]

whether the cell allocation has been successful and if yes which cells are allocated. In case of an error or package loss 6P also has mechanism in place such as squence numbers and timeouts in order to identfy and deal with errors.

6top only defines the mechanism by which the cells are allocated, but it doesn't specify when, how many and which cells are added or deleted for instance. In other words it only provides a framework to manage the schedule, but intentionally leaves the actual implementation of how to manage the schedule undefined. This is due to the different needs of different applications that prioritize different Key Performance Indicator (KPI) and therefore prefer different ways to mangage the schedule. The before mentioned SF is the entitiy that takes care of this task and beyond it's tasks is not further defined in the 6TiSCH specifications.

2.2.4 Minimal Scheduling Function (MSF)

One such SF is the Minimal Scheduling Function (MSF) [12], which is a simple and decentralized approach to managing the TSCH schedule. It is also the only SF to be defined by a RFC making it widely used and researched.

Name	RECOMMENDED value
SLOTFRAME_LENGTH	101 slots
NUM_CH_OFFSET	16
MAX_NUM_CELLS	100
LIM_NUMCELLSUSED_HIGH	75
LIM_NUMCELLSUSED_LOW	25
MAX_NUMTX	256
HOUSEKEEPINGCOLLISION_PERIOD	1 min
RELOCATE_PDRTHRES	50 %
QUARANTINE_DURATION	5 min
WAIT_DURATION_MIN	30 s
WAIT_DURATION_MAX	60 s

Table 2.2: MSF recommended parameters RFC9033. [12]

MSF defines three different types of cells for communication. First it introduces a shared cell typically in the first slot of each slotframe, which is called the minimal cell. This cell is implemented by default and is used for EBs, RPL control messages or join requests from new nodes.

Second MSF defines autonomous cells, that act as default cells to bootstrap unicast communications. Every node has a permanent Rx autonomous cell whose location in the slotframe is derived from Equation 2.2 calculating the slotOffset by Symbolic Aggregate approXimation (SAX) hashing [13] the 64-bit Extended Unique Identifier 64-bit (EUI-64) of the node and Equation 2.3 calculating the channelOffset also by hashing the 64-bit EUI-64. The slotOffset is incremented by one to avoid it interfering with the minimal cell and both hashing computations are limited by the second input given in the hash function, to not exceed the the amount of available timeslots or frequency channels. On this shared cell the children of the node then if needed can schedule Tx cells to the parent and send traffic.

$$\text{slotOffset}(\text{MAC}) = 1 + \text{hash}(\text{EUI64}, \text{length}(\text{Slotframe\_1}) - 1) \tag{2.2}$$

$$\text{channelOffset}(\text{MAC}) = \text{hash}(\text{EUI64}, \text{NUM\_CH\_OFFSET}) \tag{2.3}$$

Third there are negotiated cells which are dedicated unicast cells between two nodes that can be dynamically added or deleted according to the current traffic load. For the negotiated cells MSF has several mechanisms in place to manage the adding, deletion, relocation and synchronization. These mechanisms are governed by several parameters, of which the recommended values are listed in Table 2.2.

Crucial for the decision making of the MSF is MAX_NUM_CELLS, which defines the length of the observation window. When NumCellsElapsed, which is the number of negotiated cells that have passed, has reached the value of MAX_NUM_CELLS,

then the variable NumCellsElapsed is set to zero and the state of the network is evaluated and according to that analysis actions are taken. In order to decide whether to add or delete cells MSF evaluates the variable NumCellsUsed which keeps track of all the negotiated cells used in the period since the last reset. If NumCellsUsed is above LIM_NUMCELLSUSED_HIGH one cell is added. In case it is lower than LIM_NUMCELLSUSED_LOW one cell is deleted.

After each HOUSEKEEPINGCOLLISION_PERIOD MSF uses the Packet Delivery Ratio (PDR) of each negotiated cell to evaluate whether relocation of these cells due to interference or multipath fading is necessary. The necessity of a relocation is assesed by applying the following equation to each cell

$$PDR_{\max} - PDR_i > \text{RELOCATE\_PDRTHRES} \tag{2.4}$$

where $PDR_{\max}$ is the PDR of the cell with the highest PDR, PDR_i is the PDR of the cell we are currently evaluating and $RELOCATE\_PDRTHRES$ being the threshold parameter defined in MSF which determines when a cell is relocated. If this equation is true then the PDR of the current cell we are evaluating is too low meaning that interference on this cell is likely leading to MSF relocating that cell using a 6P relocate request.

Using MSF as scheduling function the boostrapping phase of the network works as follows. First the manually set TSCH coordinator starts up and sends periodic EBs which contain information about the schloframe length, channel hopping sequence and the ASN. Upon receiving a EB a node attempting to join synchronizes its clock to the networks schedule, aligning with the TSCH timeslot structure. Then initial slots are allocated in the node attempting to join, such as the minimal cell and autonomous cells. Communicating on these cells the node configures its preferred parent via RPL with DOI messages. Once the node has joined the DODAG it has full connectivity with the network and can now using MSF dynamically negotiate cells with its parent.

Cell allocation with sensing

In the RFC for MSF [12] the sensing cell allocation mechanism is described as an optional extention of the cell allocation mechanism. It suggests a sensing approach to the cell list compilation for the 6P add request where MSF creates and maintains a list of candidate cells that are then used as the cell list for cell allocations. On these candidate cells the MSF installs Rx cells to monitor for IEEE802.15.4 traffic and if traffic is sensed then the cell is removed from the candidate cell list and a new cell is randomly chosen.

3 Related Work

Concerning the research done in this field concerning 6TiSCH and MSF the overwhelming majority of it is either of analytical or simulation based nature. They evaluate common KPIs such as end-to-end delay, PDR or network adaptation time. Since for meaningful results in experimental validation hardware needs to be configured correctly, enough nodes arranged and many iterations done which is tedious and time consuming many researchers in order to validate their analytical models tend to turn to simulations. Since these can be automated and run in a fraction of the time of experiments they can save a lot of time but still provide support for the mathematical models.

3.1 6TiSCH Minimal Scheduling Function: Performance evaluation

In this paper [14] the authors recommend values for the MSF parameter MAX_NUM_CELLS based on simulation results for different network scenarios. The goal is to find suitable values, that increase the adaptability of the network and reduces the 6P control traffic overhead. In addition, the paper also proposes an improved version of MSF, which is implemented and tested in simulations.

For the simulator they implemented the whole 6TiSCH protocol stack in python and constructed a linear 4 hop topology using 5 motes. For each scenario the simulation was run 1000 times with values for MAX_NUM_CELLS= 4,8,16,32. As relevant Key Performance Indicators (KPI) the number of 6P messages sent by each mote and the end-to-end latency of each packet was considered, in order to evaluate which value for MAX_NUM_CELLS is the optimal.

For periodic traffic in which each mote generates one packet per slotframe the result was, that the lower MAX_NUM_CELLS was chosen, the higher the 6P control traffic overhead was. At the same time the end-to-end delay was little influnced by the change of MAX_NUM_CELLS, since MSF already prepares cells in case traffic load increases. As a result of these findings the suggested equation for periodic traffic to calculate the value of MAX_NUM_CELLS is:

$$MAX\_NUM\_CELLS \geq 2(L \div 50\%) = 4L \tag{3.1}$$

with L being the traffic load of packets per slotframe and 50% the average cell usage.

For bursty traffic the a similar pattern is observed in which the lower the value for MAX_NUM_CELLS the higher the 6P control traffic overhead with small improvements of the end-to-end delay.

Considering the amount of 6P control traffic using MSF, since it can only add or delete one cell at a time the authors suggested an advanced version of the MSF (A-MSF) where the amount of cells to be added or deleted is dynamically calculated. For the amount of cells to add the following equation is suggested:

$$\mathrm{X} = \mathrm{numTxCellScheduled}(2\mathrm{cellUsage} - 1) \tag{3.2}$$

with numTxCellScheduled being the cells scheduled in the last housekeeping period and cellUsage defined as the cells used in that same period. For the amount of cells to be deleted they suggested this:

$$\mathrm{X} = \mathrm{numTxCellScheduled}(1 - 2\mathrm{cellUsage}) \tag{3.3}$$

With these improvements the authors found that the network improves in 6P control taffic overhead for low MAX_NUM_CELLS values while maintaining the end-to-end delay.

3.2 IMSF: Improved Minimal Scheduling Function for Link Scheduling in 6TiSCH Networks

For this paper [15] the authors suggest an improved MSF where they aim to cut 6P control traffic, packet loss and end to end delay.

The authors identify two major problems of the MSF algorithm which are first, that it only has a reactive mechanism to traffic change. Meaning that the system always lags behind the actual number of cells needed for the network to be most efficient. Second the MSF is not flexible in the number of cells it adds/delets, when it senses a change of traffic. This is especially inefficient when there is a sudden spike in traffic, which would require the network to provide high amounts of cells to be added quickly.

With this in mind they suggest an improved minimal scheduling function (IMSF), which estimates the expected NumCellUsed i.e. the future traffic demand, calculates the cells needed to be added/deleted and then sends a single 6P add/remove request based on the results of the calculation.

To calculate the estimated amount of cells to add they suggest this equation:

$$0.25 < \frac{T(i,A)ETX}{16(len(Tx_CELLS) + x)} \leq 0.75 \tag{3.4}$$

with Expected Transmission Count (ETX) and .... as the queued packets already existing in that node.

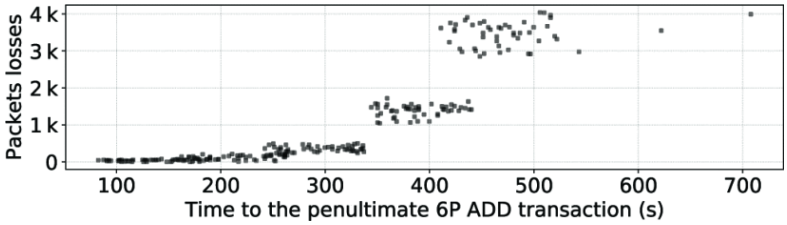

Figure 3.1: Packet loss in relation to time it takes for the constant traffic network to stabilize.

In order to calculate the estimated amount of cells to delete they use this equation:

$$y = \lfloor len(Tx_C ELLS)(\frac{0.75 - Cell_u tilization}{0.75}) \rfloor \tag{3.5}$$

To test the IMSF a python based open-source simulator for 6TiSCh networks was used to simulate three different network topologies, varying amounts of nodes and with bursty and periodic traffic. The results are, that the IMSF consistently outperforms the default MSF and A-MSF, which has been suggested in the paper [14] detailed above. Under different network conditions it can be observed, that 6P control traffic overhead and also end-to-end delay have been significantly decreased.

3.3 Pushing 6TiSCH Minimal Scheduling Function (MSF) to the Limits

In order to gain a deeper understanding on 6TiSCH MSF on a fundamental level the authors of this paper [16] have simulated a network with constant and varying traffic to figure out how effective MSF's adaptation mechanisms are. For their python based simulation [17] they implemented a 5 node linear topology with the standard recommended values for MSF. For the data collection node 2 was chosen due to the fact, that all the traffic from the nodes before pass through this node.

Firstly they examined the behaviour of the network under constant traffic and how it adapts to the initial change and then stabilizes. One of the main findings w as, that the lenght of the cell allocation period has a major affect on the PDR of the network as can be seen in figure 3.1. That is because in that period the packet queue of a node fills up, since there is not enough cells per slotframe to handle the increased traffic, and at a certain point when the queue is full packets then are dropped. Another pattern that was observed was, that with higher traffic rates the PDR increased while the latency decreased. This counter-intuitive result comes from the fact, that with more traffic also more cells are allocated making it more likely for a packet to have a close cell cell scheduled in which it can be sent immediately after arriving at a node.

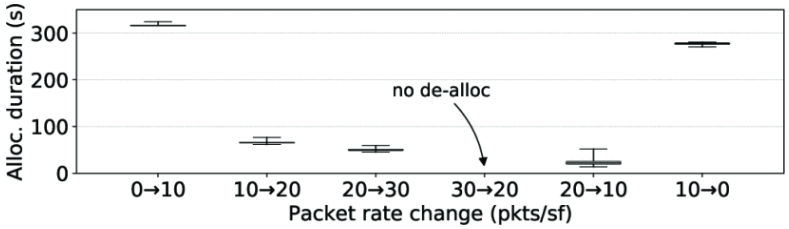

Figure 3.2: Time of adaptation for each traffic change.

Secondly a network with periodic change in traffic was simulated and observed. The main objective was to easure the time period from when the packet rate change to when the schedule reached a stable state. It took the longest in the beginning of the network, since there were little cells to send communication traffic on resulting in the allocation process to take very long. But with increased traffic the time to allocated then stayed roughly the same as can be observed in figure 3.2.

This paper gives a good first impression on the effectiveness of the adaptation mechanisms of 6TiSCH MSF and shows the importance of the cell allocation time as a key performace indicator (KPI), directly affecting the PDR.

3.4 Thorough Performance Evaluation & Analysis of the 6TiSCH Minimal Scheduling Function (MSF)

In this paper [11] David Hauweele et al. the authors of [16] mentioned in the previous section expanded their work on 6TiSCH MSF evaluation by studying the behaviour and reactivity of the network under varying traffic loads and proposing a mathematical model in oder to predict the convergence pattern of MSF. The evaluations made in the same python based simulator were conducted on a linear topology with constant and varying traffic loads. Results show that the more cells are allocated the faster MSF adapts to changing traffic. It was observed that at higher traffic loads MSF tends to overprovision while improving latency, since using the recommended values it allocates cells at a 75% cell utilization but only deallocates cells once cell utilization is below 25%. The paper propses following equation

$$T_{msf}(a,b) = T_{sf} \times \sum_{k=a}^{b-1} \left(\frac{1}{2} + \frac{1}{2k} + \frac{\text{MAX NUMCELLS}}{k} \right) \tag{3.6}$$

to calculate the time required by MSF in order to go from a to b allocated cells. The model multiplies T_{sf} which is the time for a slotframe by the necessary slotframes to allocate all cells. The latter is computed by taking the sum of all slotframes needed for each allocation.

At last the paper investigates the impact of different MSF parameters on packet losses and resource utilization. Findings include that by reducing MAX_NUM_CELLS faster traffic adaptation by quicker cell allocation can be achieved, due to the shortened period in which traffic load is evaluated. This however also makes the evaluation less precise since the amount of gathered data is less potentially leading to inefficient allocation patterns. To limit overprovisioning adjusting the thresholds for cell allocation and deallocation was found to be effective, where a smaller gap between the lower and upper bound reduced overprovisioning while increasing allocations and deallocations. This shows the tradeoff that is to be made between precision in estimating the traffic load and stability, since with the former small changes already trigger allocations or deallocations and with the latter overprovisioning is high. The paper concludes that improving MSF requires modifying the SF itself, which for example the A-MSF approach suggests, because attempts to reduce convergence time and limit overprovisioning caused instability in the network.

3.5 An Experimental Evaluation of the 6top Protocol for Industrial IoT Applications

For evaluating 6top and it's performance in a real environment [18] Francesca Righetti et al. set up a experimental testbed of nodes. They had a network of 23 nodes with each being a Zolertia RE-Mote sensor node running on Contiki OS which is a operating system for sensor nodes with built in 6TiSCH implementation. As SF they used On-the-Fly (OTF) [19], which allocates cells according to the data traffic demand and Proportional, Integral, Derivative (PID) [20], which allocates cells according to the queue occupancy.

For the experiments the authors have three metrics to evaluate the performance, which are the success rate, transaction delay and the failure rate. Applying these KPIs the team found that the SF has a big influence on how efficient the 6top layer works. For instance for the success rate of the 6P transactions the failure rate for each tested traffic rate was below 70% for PID. In contrast to that with the same traffic rates OTF manages to maintain 6P transaction failure rates above 80%. This is due to the fact that for PID a lot of allocations and deallocations are made, which increase the number of 6P transactions and therefore leading to a high failure rate. Another finding that is presented is the actual transaction delay that is defined as the time from when the 6P request is send to the time when a response is recieved. In previous papers this time has been assumed as negligible [21], but after real world testing the authors found, that depending on the traffic rate and SF average transaction delays from up to 0.5 seconds to 2.2 seconds can occour. As main contributing factors to these failures the authors list insufficient resources, 6top timeout expiration, schedule mismatch and buffer overflow.

3.6 Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation

In [22] Accettura et al. propose the decentralized traffic-aware scheduling (DeTAS) approach, which is the decentralized extention of the traffic-aware scheduling approach (TASA) proposed first by [23]. It aims to achieve collision-free schedules in a multihop TSCH network by using a small amount of information that is locally communicated between the nodes in order to configure the schedule. DeTAS takes into account the queue levels thereby avoiding traffic congestion and reducing the possiblity of packet drops due to the queue being full. This traffic information of a node is received from the children and the node itself also passes it's traffic amount up to the parent. From this information the network then can decentrally build up a collision free schedule by allocating transmission and reception cells in a way that ensures nodes with a shared communication space do not transmit at the same time therefor minimizing collisions.

To facilitat the exchange of traffic information among nodes the paper also introduces DeTAS MAC command frames. It firstly defines the REQ command, which is sent by a node to its parent with information about it's traffic requesting scheduling information from the parent. Secondly the RES command is sent as a response by the parent in order to provide the requested scheduling parameters to the child node.

With DeTAS implemented in the OpenWSN [24] project a open source implementation of standard communication protocols such as TSCH and RPL the authors used TelosB motes to experimentally validate it's performance. Two different network topologies were implemented namely a double chain topology evaluating network depth and a binary tree topology assessing performance in dense networks. As KPI end-to-end delay, link packet loss ratio (PLR) and node duty cycle were considered.

In the experiments on the double chain topology it was found that with increased distance to the root node the duty cycle decreased, since the nodes closer to the root thus acting as traffic bottlenecks while end-to-end delay increases. With a increased number of channel offsets the duty cycle, interference and end-to-end delay were able to be reduced. For the experiments on the binary tree topology similar results were observed confirming, that DeTAS performs consistently across various topologies.

The paper provides a decentralized approach to scheduling in 6TiSCH and validates it's proposal with experimental tests showing the value of an experimental approach to validating performance of scheduling approaches.

3.7 Novelty

The research into MSF has been quite deep from a theoretical point of view and many aspects such as the ideal amount of cells to add, best MSF parameters to choose for

a desired behaviour and overprovisioning have been studied in depth. Using models and validating them with simulations a good understanding of MSF has been gained also leading to suggestions of improvement such as with the A-MSF [14] or IMSF [15]. Although the cell allocation process has been part of some of the above mentioned works the mechanism for which cells to include into the cell list of a 6P add request has not been evaluated at all. In addition to that none of the MSF focused works have used experimental methods to validate their findings showing that there is a lack of experimental approaches when it comes to studying MSF. Taking that into account the following sections attempt to present a analytical model for the cell allocation time taking into account the cell list selection mechanism and integrating the sensing improvement which was suggested in the RFC. The provides a deeper understanding of the cell allocation mechanism and seeks to offer insight into what could be more effective ways of compiling a 6P add cell list. Furthermore the results are validated by conducting experiments and discussing them alongside the analytical results.

4 Analytical Model

The analytical model developed and outlined in this work assumes a two node network consisting of parent and child and additional an network with constant traffic rate. The scope of the analysis focuses on the cell allocation period and does not include the bootstrapping phase of the network. As KPI we consider the time it takes for the schedule to reach a stable state and the probability of cell overlap, since they best capture a holistic performance of the scheduling function and it's subsequent cell allocation mechanism.

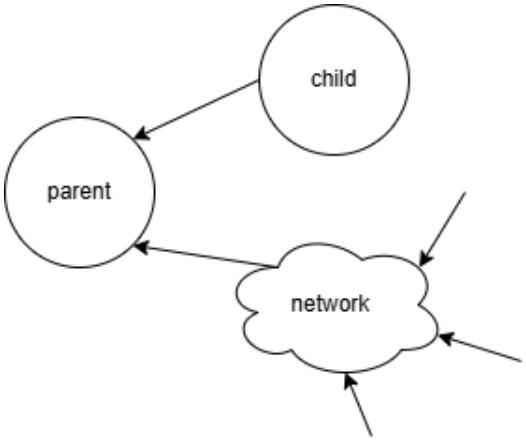

Figure 4.1: Network topology assumed by the model

4.1 Default Cell Allocation Mechanism

In order to determine how effective a certain cell allocation mechanism is we consider the time it takes for a network to reach a stable state after adapting to a constant traffic rate of λ which is being served by allocating $\mu_{\max}$ amount of cells. The network is considered stable if all cells have been evaluated in terms of PDR and none have been determined to be in need of further relocation. The cell allocation mechanism mostly affects the time it takes to allocate a certain amount of cells and the quality of the cells, meaning how much interference there is on them. For this reason using scheduling time as KPI is an effective way to measure the performance of the cell allocation mechanism, because more low quality cells also means more relocations needed, resulting in an overall longer time until the network is stable.

The time T_s it takes for a network to reach a stable state is calculated by:

$$T_s = T_a + T_r \tag{4.1}$$

where T_a is the time it takes to allocated all $\mu_{\max}$ cells and T_r is the time it takes for all of the cells to be relocated until there is no relocation necessary.

4.1.1 Cell Allocation Time

The time to allocate $\mu_{\max}$ cells is calculated as follows:

$$T_a = \sum_{i=2}^{\mu_{\max}} \left(\frac{M}{i-1} + \frac{1}{i} + 0.5 \right) \tag{4.2}$$

where we take the sum of the time it takes to allocate each cell until we reach $\mu_{\max}$ cells, which is derived from the target traffic rate λ (packets per SF) defined by the application and is calculated as:

$$\mu_{\max} = \left\lceil \frac{\lambda}{u_{high}} \right\rceil$$

with u_{high} being the upper cell utilization threshold in MSF which determines at what cell utilization threshold another cell is added. For this model we start at the second cell since the mechanism for the allocation of the first cell in MSF is a special case that counts autonomous cells into the NumCellsElapsed which normally only keeps track of negotiated cells and not autonomous cells. So for the sake of simplicity and generality we leave out the allocation of the first cell and consider it from there onwards.

For each cell allocation we first need to wait for MSF to recognize that a cell allocation is necessary. This happens every time the number of elapsed cells reaches MAX_NUM_CELLS. By dividing MAX_NUM_CELLS by the amount of cells already scheduled we can get the time it takes for MSF to initiate another cell allocation.

Secondly we need to factor in the time it takes for the 6P ADD request to be sent and for the response to be received. Considering [25] and adapting the models to our case the second addend models the mean waiting time to the next scheduled cell with $\mu_i - 1$ cells, since μ_i is the cell which we are trying to allocate but hasn't been allocated yet. At last the third addend in this sum accounts for the waiting time of the 6P RESPONSE to the next scheduled cell in the parent node. This stays constant, since we assume that the parent node only communicates via the autonomous cell to the child node.

4.1.2 Cell relocation time

Assuming recommended values for the MSF SF presented in Table 2.2 the cell relocation time is calculated as

$$T_r = t_h \min(\lfloor E_\Sigma[O] \rfloor, 1) + \left(\frac{1}{\mu_i} + 0.5\right) \left\lceil \frac{\lfloor E_\Sigma[O] \rfloor}{r_l} \right\rceil \tag{4.3}$$

where the first addend models the time it takes for MSF to evaluate the cell performance and, if needed, initiate a cell relocation. The second addend models the time it takes for the 6P RELOCATE requests and responses to be exchanged.

In order to calculate t_h, which is the time it takes for MSF to evaluate all cells we can use the following equation:

$$t_h = \left\lceil \frac{MAX\_NUMTX t_{slotframe}}{t_{housekeeping}} \right\rceil t_{housekeeping} \tag{4.4}$$

Since only after a cell has been used for MAX_NUMTX times and the value has been halved it will be evaluated for relocation we multiply the amount of times the cell needs to be used by the time it takes for a slotframe to pass $t_{slotframe}$. Then we take the rounded up value of that product and divide it by $t_{housekeeping}$ which is the HOUSEKEEPINGCOLLISION_PERIOD. This yields the number of HOUSEKEEPINGCOLLISION_PERIODs we need to wait for the cells PDR to be evaluated for relocation. To obtain the actual time we then at last multiply it by $t_{housekeeping}$.

To model the likelihood of a relocation to occur we first consider the probability from a node's perspective for a cell that it is about to allocate to overlap with the cells that already are occupied by neighbors in its communication range. For a network with constant traffic of N cells per slotframe and X being the total amount of cells available in the network the probability of a cell overlap of the i-th cell allocated is calculated by:

$$p_{ov}(\mu_i) = \frac{N}{X - \mu_{i-1}}, \quad X = n_{ch} n_{sf}, \tag{4.5}$$

where X is computed by the product of available channels and timeslots in a slotframe.

In order to quantify from this how many relocations on average are needed we can calculate the sum of expected cell overlaps like in Equation 4.2 from the second allocated cell to $\mu_{\max}$ by

$$E_\Sigma[O] = \sum_{i=1}^{\mu_{\max}} \frac{p_{ov}(\mu_i)}{1 - p_{ov}} \tag{4.6}$$

Assuming that if multiple relocations are necessary they are all done in the same HOUSEKEEPINGCOLLISION_PERIOD we only need to consider this value until it is >= 1, which is also the reason why for Equation 4.3 we take the minimum of $E_\Sigma[O]$ and 1.

The second addend in Equation 4.3 calculates the time it takes for the individual requests to be sent and the respective response to be received. Similarly to calculating the cell allocation time we can rely on the work done in [25] to estimate the time for these messages to be sent. In order to account for multiple 6P RELOCATE requsts we multiply the time it takes for the messages to be sent by the rounded up value of $E_\Sigma[O]$ the expected amount of relocations divided by r_l - the limit of relocations per message. To calculate r_l we consider the MTU of IEEE 802.15.4 and the standards defined by [9] regarding the 6P RELOCATE request:

$$r_l = \left\lfloor \frac{P_{\max}}{(\eta+1)c} \right\rfloor, \quad \eta \geq 1 \tag{4.7}$$

With $P_{\max}$ being the the available payload size without header in bytes, c the size of a cell in bytes and η the redundancy constant which ensures that for each cell relocated at least two candidate cells are included. Plugging in the standard values for a 6TiSCH network and with a minimum η of 1 we get a maximum of 12 cells per relocation request.

4.2 Cell Allocation Mechanism with Sensing

The sensing approach studied here is based off the suggestion from [12] described in Section 2.2.4, but with some adjustments. The cells in which traffic is sensed and that are being removed from the candidate cell list will be put on a blacklist to prevent them from being chosen again. In this manner we have a list of cells which we are confident in having next to no interference and a list of cells where we know that there is interference.

With this sensing approach we now only consider the probability of overlap for the cells in the candidate list, since they are the only ones used to compile the cell list of the 6P add request. This set of candidate cells C is chosen from among the pool of all available cells X'. Considering this we can calculate the probability of choosing cells in a way that at least one cell is overlapped with following equation:

$$p_{ov}^{(C)} = 1 - (1 - \frac{N}{X'})^C, \quad X' = X - n_{min} - \mu_i - n_{auto} \tag{4.8}$$

where C is the amount of cells in the candidate cell list and X' being calculated by the total amount of cells available minus the amount of scheduled minimal cells n_{min} the amount of already negotiated cells μ_i and the amount of scheduled autonomous cells n_{auto}. The probability of at least one cell in the C candidate cells being overlapped $p_{ov}^{(C)}$ is computed by one minus the probability of all cells chosen not being overlapped.

Over time with sensing the cells and replacing the ones with sensed overlap the list of candidate cells will be overlap free. The time it takes to sense all the cells in the candidate list t_s is determined by the amount of sensing operations done and the amount of sensing operations needed to sense all the overlapped cells. The minimum amount of sensing operations needed to ensure no overlap is computed by

$$n_{s\_min} = \lceil Cp_{ov}^{(C)} \rceil \zeta \quad , \tag{4.9}$$

where ζ is the redundancy factor describing how many times a cell must be sensed with no traffic detected in order to have the confidence that it is not overlapped. Similarly we can also compute the maximum and the average amount of sensing operations needed in order to have confidence in all C candidate cells.

$$n_{s\_max} = C\zeta \tag{4.10}$$

$$\bar{n_s} = C(1 - \frac{1}{1 + \lceil Cp_{ov}^{(C)} \rceil})\zeta \tag{4.11}$$

Using this we then can calculate the time it takes on average for the candidate cell list to be considered overlap free by

$$\bar{t_s} = \frac{\bar{n_s}}{X_s} \quad , \tag{4.12}$$

where X_s is the amount of cells sensed per slotframe. Considering that using recommended MSF values the time it takes to allocate a cell is in the order of tens to hundreds of seconds with reasonable X_s the time $\bar{t_s}$ becomes negelectable. This means that to integrate the sensing mechanism into the model previously introduced in Equation 4.1 we simply assume p_{ov} used in Equation 4.6 to be zero at the time of cell allocation.

The fact that cells allocated with the sensing mechanism have a higher PDR leading to less 6P messages being lost and therefor decreasing the time wasted on waiting for 6P timeouts and retries was considered. But due to time constraints and the limited scope of this work its effect on T_s was not included in the mathematical model.

5 Experimental validation

This chapter goes into detail about the experimental setup, execution of the experiments and a discussion of the results obtained in the experiments compared to the calculations of the analytical model.

5.1 Methodology

In order to experimentally validate the performance of cell allocation mechanisms a setup has been implemented, which models the network topology shown in Figure 4.1 in which there are two nodes and a neighboring network with constant traffic rate. This network with constant traffic rate is necessary in order to cause cell overlaps resulting in relocations. The coming sections will outline the technologies used to emulate this setup and the methods used for collecting and processing the data.

5.1.1 Openmote-B and Contiki-NG

As hardware platforms for the experiments we chose the Ultra Low-power Openmote-B board, which has been designed specifically for developing and researching low power wireless networks in the domain of Industrial Internet of Things (IIoT). The board is based on the Texas Instruments CC2538 System on Chip (SoC) with integrated 32-bit ARM Cortex-M3 microcontroller. It allows for wireless communication on 2.4GHz and Sub-GHz frequencies via external antennas fully supporting the latest IEEE 802.15.4 standard and features a serial USB-Port connection for flashing and logging.

As operating system for the Openmote-B boards we used the Contiki-NG (Next-Generation) real-time operating system (RTOS). It is a continuation of the popular Contiki-OS improving it in performance, and usability while staying lightweight. As a open-source operating system it is primarily designed for resource-constrained devices most commonly found in IoT with limited memory, processing power and battery life. The default version of Contiki-NG comes with widespread IoT protocols implemented such as RPL, IPv6/6LoWPAN and 6TiSCH. It's compatibility with a wide array of platforms such as the Openmote or the Zolertia Zoul makes it a popular choice for developing low-power wireless applications. One of the most unique features of Contiki-NG is the Cooja network simulator that allows for simulation of networks using the above mentioned supported

platforms running the Contiki-NG operating system without having the actual hardware. It allows researchers and developers to monitor communication, logs and energy consumption without having to deploy an actual physical setup of the network.

At the core of Contiki-NG lies a event driven system enabled by a event loop that constantly checks for events and then yields control to the process associated with that event. This approach using so called Protoheads as lightweight and stackless threads implements cooperative multitasking. It means that each process voluntarily yields back control to the scheduler. This allows for efficient and low power multitasking without the overhead of context-switching operations usually deployed by preemptive systems. Applications are implemented in such processes, which become active if the respective event is triggered and then perform the tasks necessary before handing back control. In Contiki-NG different types of timers provide precise tools for time sensitive applications. Whether the event timer for periodic tasks or the real time timer for high precision timing Contiki-NG provides a simple timer API to initialize, start, stop, reset and read timers. When getting the current system time it returns ticks, which are the CPU clock cycles since the start of the system. In order to obtain the real world time the platform specific CLOCKSECONDS variable can be used which is the ticks per seconds for each platform.

Natively Contiki-NG supports IEEE 802.15.4 and already provides an implementation of the TSCH schedule and 6top-sublayer. In order to implement the modifications necessary for conducting the experiments we need to first have a rough understanding of the inner workings of the 6top implementation in Contiki-NG.

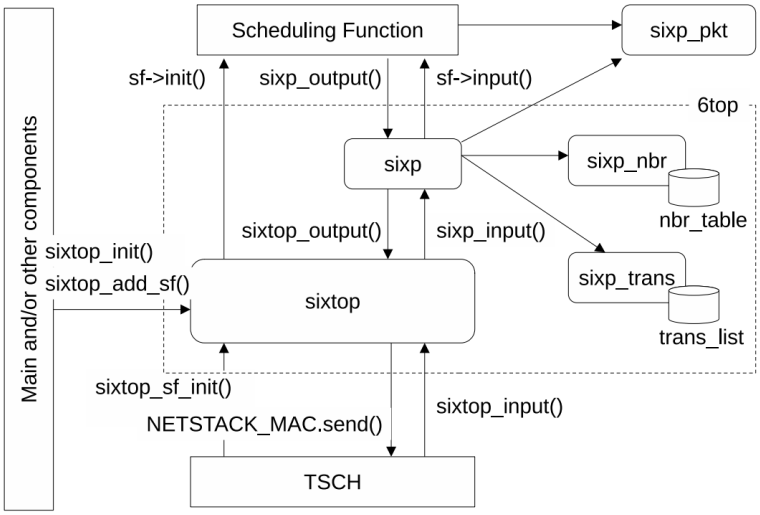

Figure 5.1: 6P implementation in Contiki-NG. [10]

From this graph it can be seen, that 6top and it's subsequent sixp module mainly provide SF with an interface for outputting 6P messages via the sixp_output() function. It keeps

track of the neighbors and manages the status of the 6P transaction ensuring, that only one is currently active. Additionally when a input is passed down from the MAC layer it calls the input function defined by the SF.

5.1.2 Data collection

The data collected for the evaluation was mainly the timing of each cell allocation, cell relocation and information about which cell was relocated. To obtain this data the serial logging function of the Openmote-B boards was used to log the outputs of the cell allocating node. The relevant data was first extracted by a python script and written into a csv format and then further processed and visualized into graphs using a second python script.

5.2 Experimental Setup and Application Design

In order to experimentally validate the analytical model we matched the topology presented in Figure 4.1 by setting up three Openmote-B nodes that each fulfill one of the distinct roles of that network. In this section the setup will be introduced and the technical details roughly explained. An exemplary view of the experimental setup can be seen in the Figure 5.2. Each experiment was conducted 10 times manually resetting the boards for a new iteration.

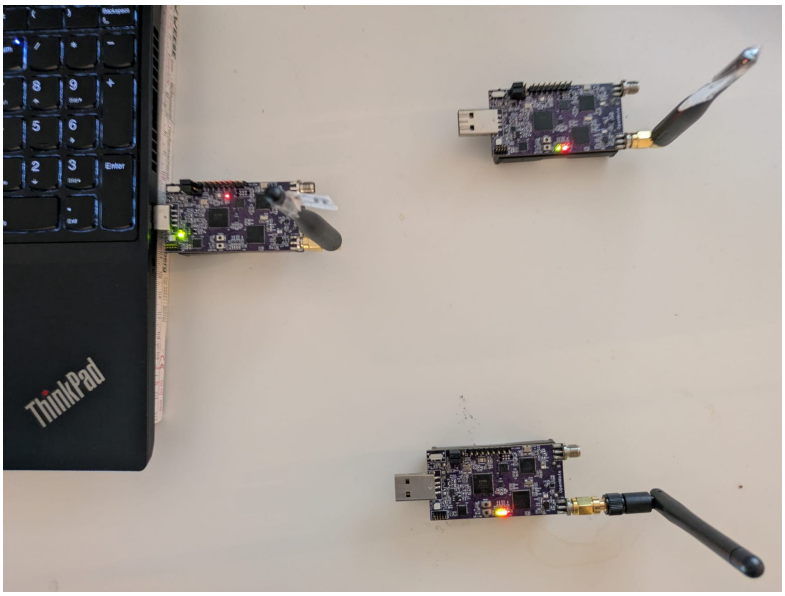

Figure 5.2: Experimental setup.

The processes on each node are based on a combination of two preinstalled examples of Contiki-NG. The first is the rpl-udp example containing the implementation of a udp client node and a udp server node able to communicate with each other via UDP messages using RPL routing. The second is the sixtop example, which comes with a rudimentary scheduling function handling the addition and deletion of cells. This includes generating and populating requests and responses but also processing incoming ones and taking appropriate action by scheduling or deleting cells in the schedule. For these tasks the TSCH API and the 6P API were used.

Parent Node

The parent node acting as the TSCH coordinator and RPL root node is running two processes. The first them sets up the tsch network and responds to any incoming 6top traffic. The second process initializes a User Datagram Protocol (UDP) server to process any incoming UDP packets. If a message is received the parent node checks whether it is from the network emulator or the child and then if no autonomous cells have been added to that node it will add them. These autonomous cells are set up in order to emulate the autonomous cells in MSF with the difference that the timeslot and channeloffset of these cells are hard coded and not calculated using the standard defined in Equation 2.2 and 2.3.

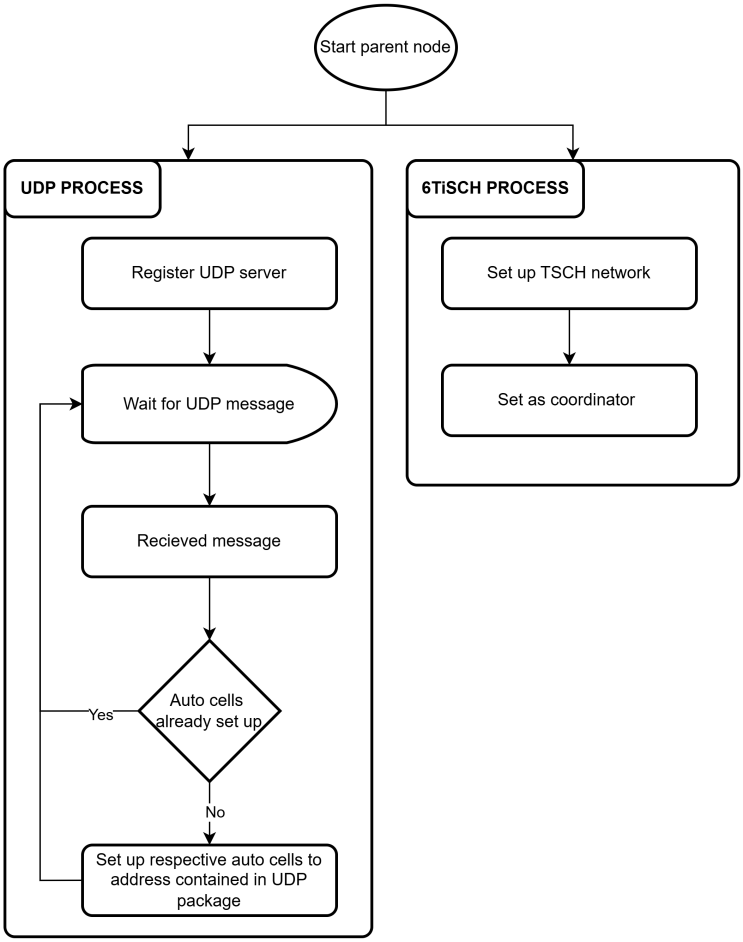

Figure 5.3: Flowchart of parent nodes actions.

Child Node

The child node is the target node, which allocates cells for communication with the parent and, if there is interference, relocates the cells. To emulate the real life scenario as much as possible the child node also sends UDP packages to the parent with a an interval of

$$t_{send} = \frac{CLOCK\_SECOND}{n_{tx}u_c} + t_{var} \tag{5.1}$$

where t_{send} is the time in clockticks between each package. It is computed taking $CLOCK\_SECOND$, which is a physical second in platform specific clockticks and dividing it by n_{tx} the amount of currently allocated cells multiplied by u_c which is the cell

utilization factor. Finally t_{var} as small variance variable is added at the end, to ensure all cells are used

All these three functionalities translate directly into their respective processes which are presented in Figure 5.4. The UDP process is responsible for sending the UDP packets to the parent with the above mentioned frequency. The 6P add process first initializes TSCH and joins the network. Upon successful joining of the node to the network the process adds the autonomous cells to the parent and logs the time of the start of the experiment. Then cell additions take place till the the target amount of cells per slotframe is reached where it then terminates. The time until MAX_NUM_CELLS have elapsed is estimated by dividing MAX_NUM_CELLS by n_{tx} and multiplying that with the slotframe length in seconds. At last the 6P relocate process waits for the node to fully join the network and then evaluates each HOUSEKEEPINGCOLLISION_PERIOD whether relocations are necessary. In case relocations are necessary the node compiles a list of cells that need to be relocated then sends a relocation request for each cell.

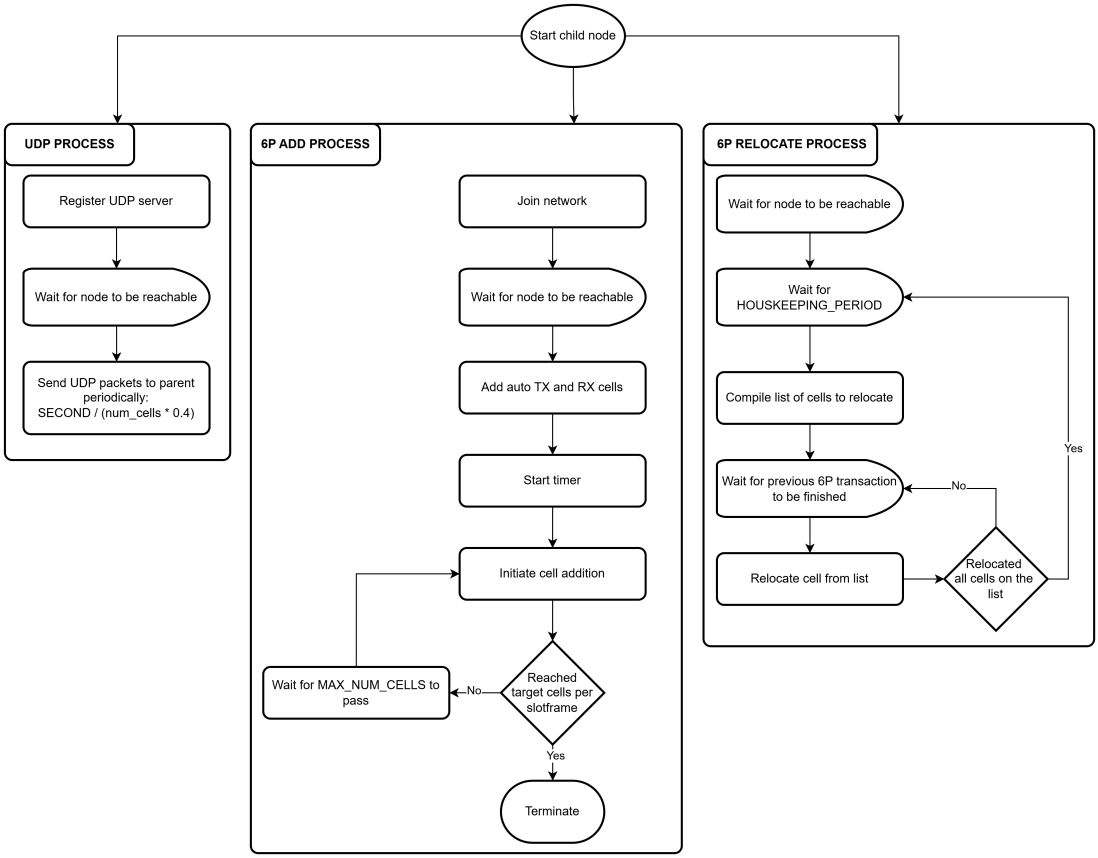

Figure 5.4: Flowchart of child nodes actions.

Network Node

The network node has the function to act as the emulator for the rest of the 6TiSCH network with a constant traffic load N packets per slotframe. To achieve that the cell initialization process of the node first joins the network and then randomly generates a given num_cells_interfere amount of cells to be allocated for interference. Once the node has joined the network these cells are added to the schedule manually, meaning without negotiating and sending 6P requests and upon adding all cells, the process terminates. To interfere on the allocated cells the UDP process first sets up a UDP server and upon successfully joining the network autonomous Tx cells to the parent are allocated to enure stable communication for RPL and TSCH control traffic. If interference is set to false then the node sends a UDP package every second in order to allow the child node to successfully join the network without interfering with that too much, since the bootstrapping period of the network is not of interest for this research. Once the child node is connected to the network the parent will send a UDP package to the network node informing it to start interferen upon which the node will start interfering with a periodicity of num_cells_interfere packets per second. These packets are UDP broadcast with the nodes link layer address as payload.

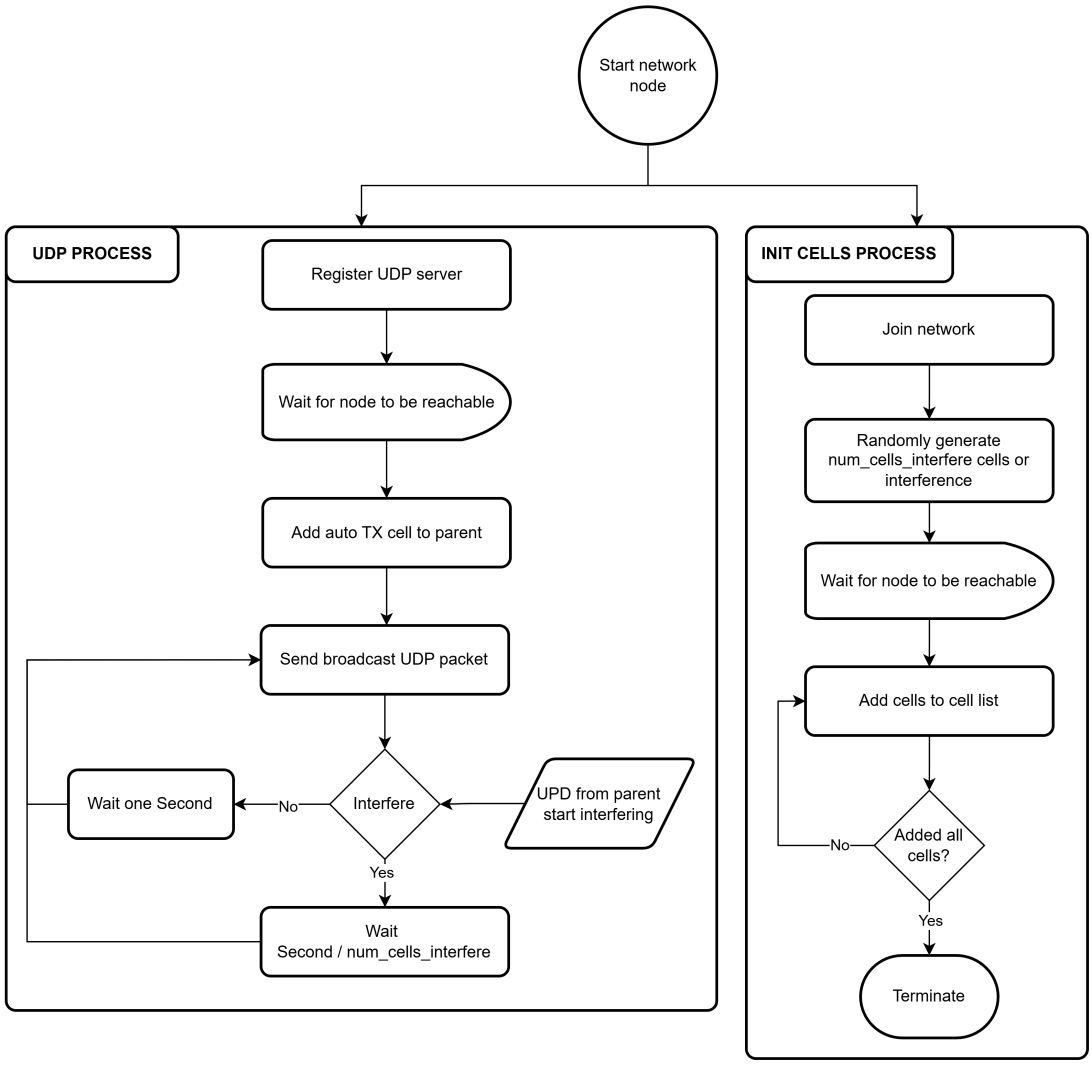

Figure 5.5: Flowchart of network nodes actions.

Additional Implementations

Since the simple SF provided by the sixtop example only has cell additions and deletions, a relocation process needed to be implemented. For the SF to be able to decide whether a relocation is necessary or not, it needs to have a precise information for every cell how many packets were sent, and how many of them were successful. The TSCH API only supports the evaluation of whole channels meaning that only the performance of a channel was recorded and could be queried. That's why an additional mechanism was added which everytime a package is sent will keep a record of which cell it was sent on and whether it was successful or not. This was the relocation process can accurately

calculate the PDR of each cell and decide whether a cell needs to be relocated or not. The relocation mechanism is designed similarly to the add and delete process. First all the entries of the cells are evaluated and the PDR is calculated. Upon discovering a PDR that is below the threshold a random candidate cell list is generated, which then together with the cell that is to be relocated and some metadata is inserted into a 6P relocated package and then sent via the 6top output API. Receiving this 6P relocate request the parent node evaluates the candidate cell list and picks the first that is available. It then sends a 6P response with the chosen cell, allocates that cell and deletes the cell that is relocated from its schedule. At last after receiving the 6P request the child also adds the chosen cell and deletes the relocated cell.

The sensing mechanism was implemented using a candidate list and a blacklist. In addition to that a randomly pre-generated list of cells was defined as cells in which the network node would interfere. Using these three lists the sensing mechanims would check before each cell allocation whether the cells in the candidate list were on the interference list. This implementation was chosen in order to emulate the sensing since the implementation of the sensing cells is not viable for the timeframe given.

5.3 Results and Discussion

This section, based on the experimental setup and application design explained before presents the results of the experiment and compares them with the results of the analytical model proposed in section 4. For this comparison two KPIs are taken into account namely the scheduling time, which is the time it takes for the schedule to reach a stable state T_s and the probability of a cell that is allocated to overlap with an occupied cell $p_o v$. The common parameters of the experiments are presented in Table 5.1. In total 8 experiments were conducted with each 10 runs allocating 25 cells in one run with 4 using the default cell allocation mechanism and 4 the sensing mechanism. For each i-th cell T_s was tracked, recorded and logged in the end of the experiment enabling us to have T_s, T_a values and the amount of relocations for $\mu_{\max}$ from 2-25 cells. The results presented are the average of all the runs with a confidence interval of 95%. The network interference is evaluated on 20% and 10% due to the limitations of using only one node as emulator for the network, since physically it at most can only send on one cell each timeslot and experimentally it turned out that a higher load leads to inconsistency in the nodes performance.

Scheduling time

First we consider the scheduling time, since it is a effective KPI in determining the performance of the cell allocation mechanism, because it reflects both efficiency and and quality of the cell allocation mechanism. It is to be noted, that for the experimental approach we measured the point in time where no allocation was made whereas the analytical model gives the time the last relocation was done. In order to accout for

Table 5.1: Common parameters of all experiments.

MAX_NUMTX	32
HOUSEKEEPINGCOLLISION_PERIOD	60s
NUM_CH_OFFSET	4
SLOTFRAME_LENGTH	101 slots
RELOCATE_PDRTHRES	50%
Number of runs	10

this discrepancy one an subtract one HOUSEKEEPINGCOLLISION_PERIOD from the experimental results. For the first experiment the following parameters were used.

Table 5.2: Paramters for the first experiment.

MAX_NUM_CELLS	100
Network interference N	20%

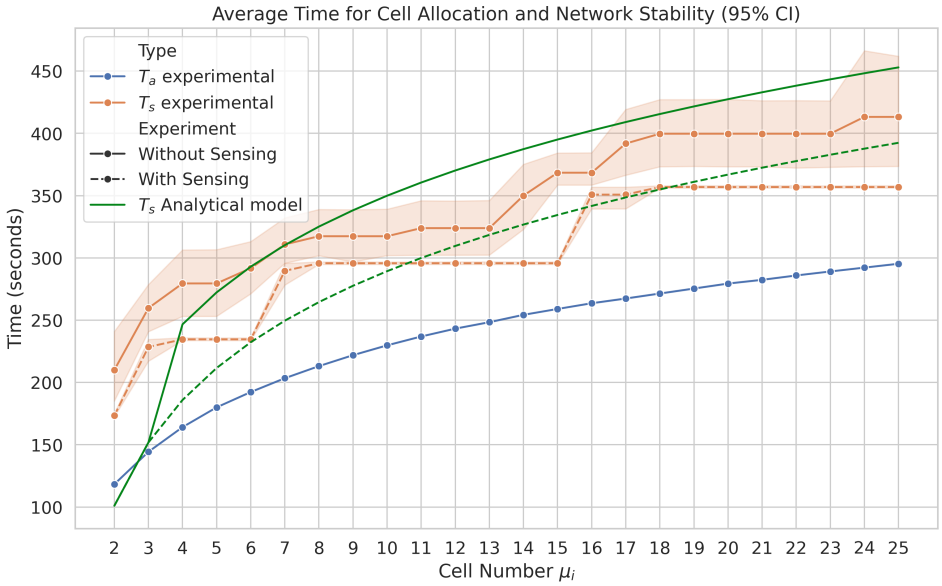

Figure 5.6: Experimental and analytical results with medium interference.

Using the recommended value for MAX_NUM_CELLS and a network interference of 20% i.e. 80 cells interfered out of 400 total available cells Figure 5.6 plots the experimental results with the expected analytical outcomes. For the T_a obtained from the experiment we can observe that with increasing $\mu_{\max}$ the time it takes to allocate an additional cell decreases since MAX_NUM_CELLS is reached faster leading to a shorter time until

another cell is added. Since the implementation uses a timer to estimate the time it takes for MAX_NUM_CELLS to be reached by NumCellsElapsed and the fact that the variation of 6P add requests is minuscule the result is a smooth but slowing increase in time it takes to allocate the cells. Considering T_s the sensing mechanism consistently outperforms the non sensing mechanism as predicted by the analytical model presented in Equation 4.1. The experiments roughly match the shape and values of the analytical model while only differing significantly in their sudden increases. The stepwise increase of the experimental data is due to the fact that if another relocation is needed it takes another HOUSEKEEPINGCOLLISION_PERIOD to relocate again and the smooth graph for the analytical data is because it works with averaged values. The variance of the non sensing mechanism is notably higher than the sensing mechanims since the sensing mechanism has next to no overlap leading to at most one relocation whereas the amount of relocation periods necessary for the non sensing mechanism can vary highly. The reason for that is for instance the fact, that the time it takes to register a cell as having a PDR that is too low can vary since even if the cell is interfered with it might not be in every slotframe, so some messages do get sent successfully.

To study the allocation mechanism under different network interference loads the second experiment had a network interference of 10%.

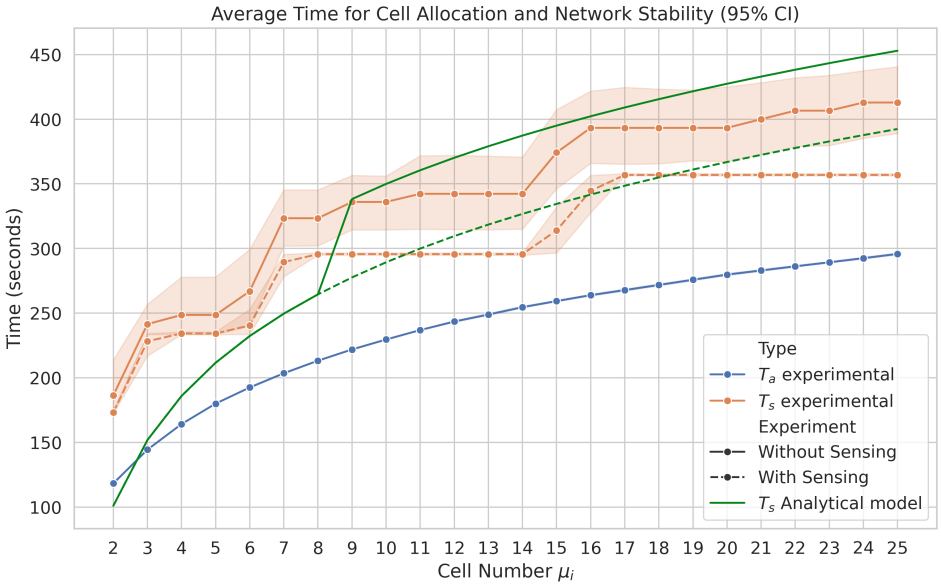

Figure 5.7: Experimental and analytical results with 10% network interference.

Similarly to the first experiment the sensing mechanism consistently outperforms the non sensing mechanism and both align with the analytical expectations. Overall compared to an interference of 20% T_a remains unchanged and T_s only is slightly lower to about a

$\mu_{\max}$ of 7 for the experimental and 8 for the analytical before it becomes very similar. This is because in the beginning the likelyhood of a relocation occuring at all very low, but the more cells are allocated the more it grows until it reached a point where at least one relocation must be done, where the amount of relocations then does not play that big of a role anymore. This is why with a lower interference the point where the sensing and non sensing mechanims diverge in terms of T_s is later at 8 cells whereas for medium interference they already diverge at 3 cells. The probability of overlap starts lower and increases slower, making the non sensing mechanism behave closer to the sensing one for longer.

In addition to varying the network interference we also varied the MSF parameter MAX_NUM_CELLS to study the effect of faster cell allocation on the cell allocation process.

Table 5.3: Paramters for the third experiment.

MAX_NUM_CELLS	50
Network interference N	20%

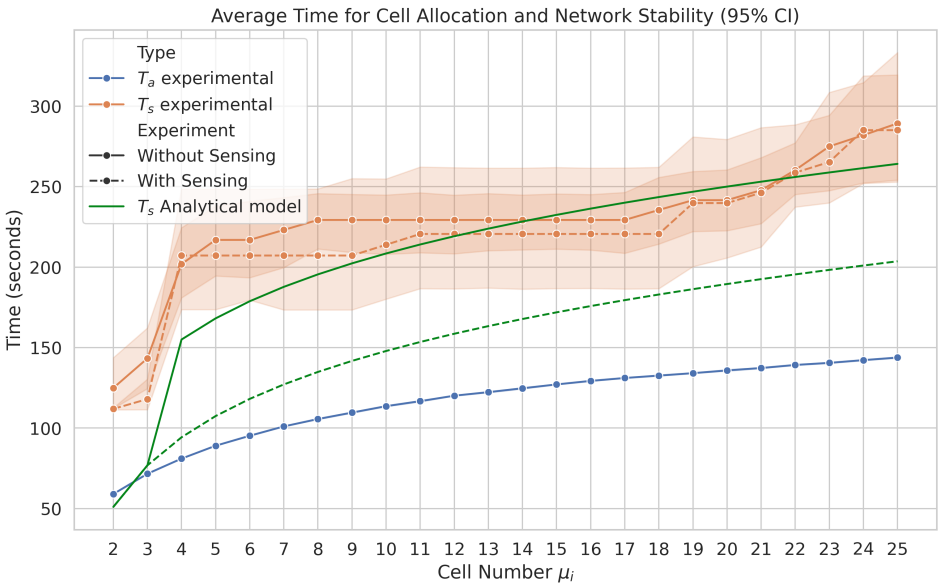

Figure 5.8: Results of low MAX_NUM_CELLS with medium interference.

As expected T_a significantly decreased from over 100 seconds with MAX_NUM_CELLS of 100 to about 60 second for MAX_NUM_CELLS of 50 in the second cell and this relationship is consistent for all cells. This is because it takes less time for NumCellsElapsed to reach MAX_NUM_CELLS leading to more frequent cell allocations. As for T_s it

also decreased by up to 100 seconds compared to before. Although the non sensing experimental values align well with the analytical expectations the values from the sensing mechanism are not as distinced from the non sensing mechanism as before. Due to experimental fluctuations with external interference?

At last we consider the network under low network interference with low MAX_NUM_CELLS.

Table 5.4: Paramters for the fourth experiment.

MAX_NUM_CELLS	50
Network interference N	10%

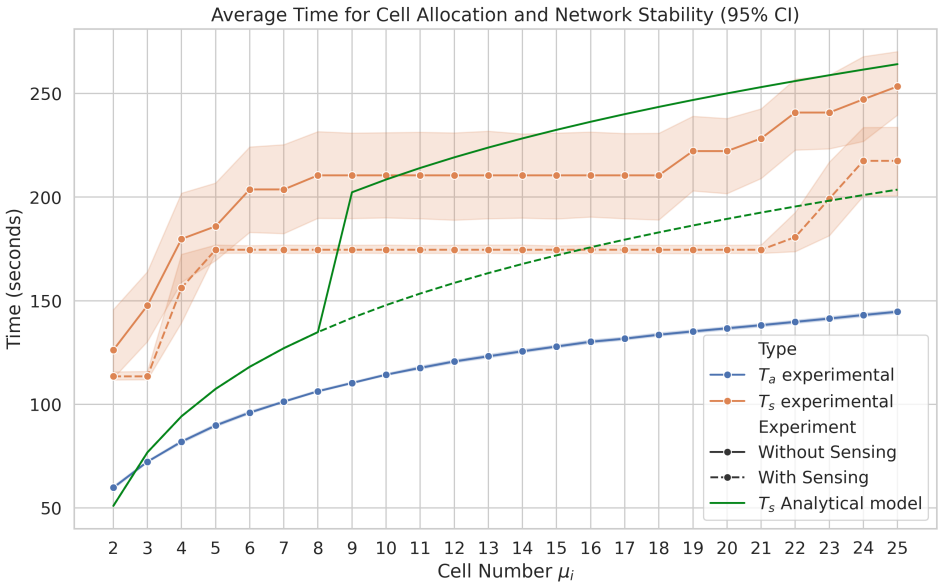

Figure 5.9: Results of low MAX_NUM_CELLS with low interference.

Compared to Figure 5.7 with the same network interference but higher MAX_NUM_CELLS the values generally are lower as we expected, due to a higher frequence of cell allocations. But the overall pattern is similar with the exception that the value plateaus at the same value from about 5 cells to 21 cells for the sensed mechanism. This is due to the mechanims of the HOUSEKEEPINGCOLLISION_PERIOD which starts simultaneously with the cell allocation mechanism and therefore no matter how many cells are allocated if the schedule stabilizes within the same HOUSEKEEPINGCOLLISION_PERIOD it will be measured to be stable at the same time.

Probability of Overlap

To further analyse the results we look at p_{ov} as KPI. From the experiments the amount of relocations for each iteration was collected and compiled for plotting. The following graphs compare p_ov for different network interferences and under varying MAX_NUM_CELLS. Each value is computed from the sum of all the relocations of each cell over each iteration of an experiment and then averaged. This approach is viable since p_ov only slightly varies depending on the i-th cell that is allocated.

First we consider p_ov with MAX_NUM_CELLS being 100 comparing low and medium network interference.

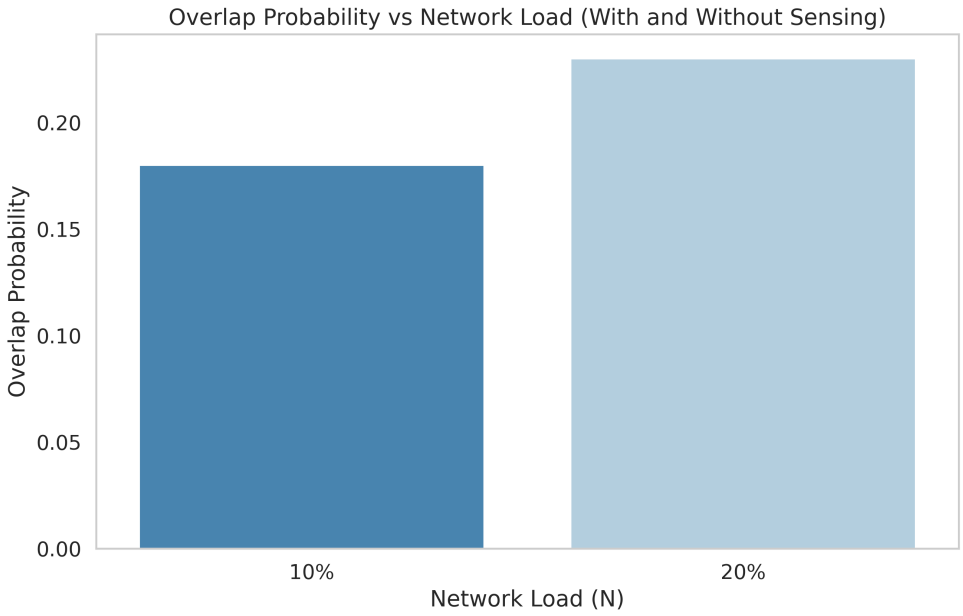

Figure 5.10: Experimental p_ov with MAX_NUM_CELLS 100 comparing low and medium network interference.

As expected with lower network interference the probability of overlap sinks, since accoring to Equation 4.5 the amount of cells occupied N goes down. For the sensing mechanism as explained in Section 4.2 the expected p_ov is zero which was also the result of the experiments. It is to be noted that this result is probably due to the approximative nature of the implementation described in Section 5.2 where the interfered cells were already known and the sensing operation was conducted with a success rate of 100%.

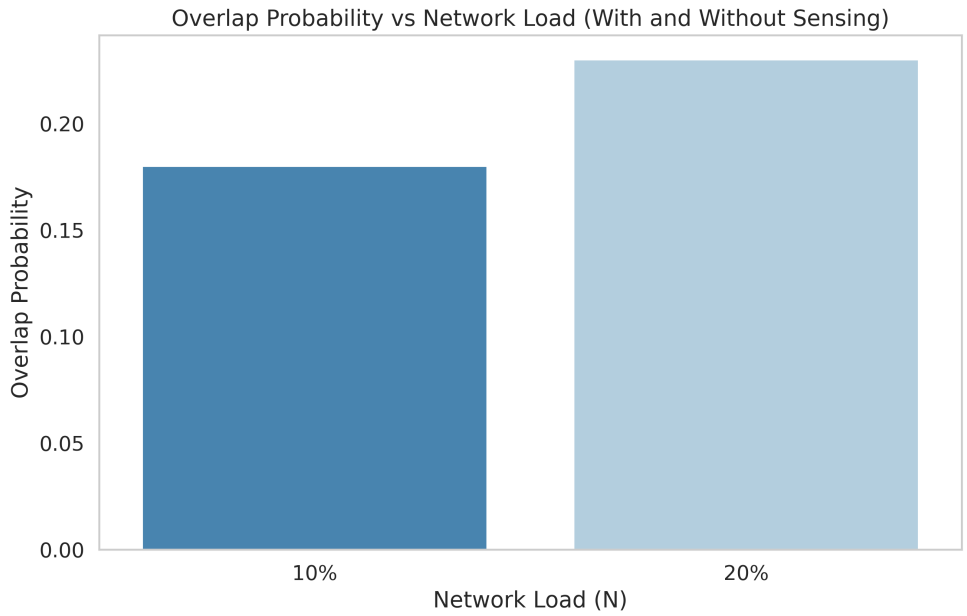

Figure 5.11: Experimental $p_o v$ with MAX_NUM_CELLS 50 comparing low and medium network interference.

For a MAX_NUM_CELLS of 50 Figure 5.11 plots the $p_o v$ for the sensing and non sensing mechanism. The probability of overlap is almost the same with the only difference being that for the sensing mechanism for a network interference of 20% we have a slight probability of overlap. This could be just as in Figure 5.8 due to external interferences causing relocations to occour.

6 Conclusion

In this work the two cell allocation mechanism were studied with varying network interference and MSF parameters. Using an analytical model and experimental validation results show that the sensing mechanism leads to less cell overlaps resulting in less time needed to allocate a certain amount of cells and for no relocations to be necessary. It was also observed that the network interference also plays a role in cell allocation time no matter the allocation mechnism. This work provides an analytical model to predict the time for a cell allocation and experimentally validates it showing the integrity of the model and also proving the effectiveness of experimental validation. Although the emulation of the sensing mechanism and network interference allowed for a meaningful analysis of the cell allocation mechanisms due to simplifications some effects were not taken into account. As further work it is suggested to fully implement the sensing with cells that sense the medium for traffic, in order to find out how effective the sensing is in detecting cells that are occupied and how fast it can compile a candidate cell list without overlap. In a addition further work can focus on refining the analytical model by considering relocations of relocated cells and the effect the sensing mechanims has on 6P timout frequencies. This work has demonstrated that the sensing cell allocation mechanism serves as a straightforward and simple enhancement to MSF, bringing improvements in various aspects.

List of Figures

List of Tables

Bibliography

[1] 'IEEE Standard for Low-Rate Wireless Networks'. In: *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)* (2020), pp. 1–800. DOI: 10.1109/IEEESTD.2020.9144691.

[2] '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)* (2012), pp. 1–225. DOI: 10.1109/IEEESTD.2012.6185525.

[3] Xavier Vilajosana, Qin Wang, Fabien Chraim et al. 'A Realistic Energy Consumption Model for TSCH Networks'. In: *IEEE Sensors Journal* 14.2 (2014), pp. 482–489. DOI: 10.1109/JSEN.2013.2285411.

[4] Pascal Thubert. *An Architecture for IPv6 over the Time-Slotted Channel Hopping Mode of IEEE 802.15.4 (6TiSCH)*. RFC 9030. May 2021. DOI: 10.17487/RFC9030. URL: https://www.rfc-editor.org/info/rfc9030.

[5] Pascal Thubert and Jonathan Hui. *Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks*. RFC 6282. September 2011. DOI: 10.17487/RFC6282. URL: https://www.rfc-editor.org/info/rfc6282.

[6] Gabriel Montenegro, Jonathan Hui, David Culler et al. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks*. RFC 4944. September 2007. DOI: 10.17487/RFC4944. URL: https://www.rfc-editor.org/info/rfc4944.

[7] Dominique Barthel, JP Vasseur, Kris Pister et al. *Routing Metrics Used for Path Calculation in Low-Power and Lossy Networks*. RFC 6551. March 2012. DOI: 10.17487/RFC6551. URL: https://www.rfc-editor.org/info/rfc6551.

[8] Sabah Suhail, Mohammad Abdellatif, Shashi Pandey et al. *Provenance-enabled Packet Path Tracing in the RPL-based Internet of Things*. November 2018. DOI: 10.48550/arXiv.1811.06143.

[9] Qin Wang, Xavier Vilajosana and Thomas Watteyne. *6TiSCH Operation Sublayer (6top) Protocol (6P)*. RFC 8480. November 2018. DOI: 10.17487/RFC8480. URL: https://www.rfc-editor.org/info/rfc8480.

[10] Yasuyuki Tanaka, Toshio Ito and Fumio Teraoka. '6TiSCH Scheduling Function Design Suite founded on Contiki-NG'. In: *Journal of Information Processing* 30 (2022), pp. 669–678. DOI: 10.2197/ipsjjip.30.669.

[11] David Hauweele, Remous-Aris Koutsiamanis, Bruno Quoitin et al. 'Thorough Performance Evaluation and Analysis of the 6TiSCH Minimal Scheduling Function (MSF)'. In: *Journal of Signal Processing Systems* 94 (January 2022). DOI: 10.1007/s11265-021-01668-w.

[12] Tengfei Chang, Mališa Vučinić, Xavier Vilajosana et al. *6TiSCH Minimal Scheduling Function (MSF)*. RFC 9033. May 2021. DOI: 10.17487/RFC9033. URL: https://www.rfc-editor.org/info/rfc9033.

[13] M. V. Ramakrishna and Justin Zobel. 'Performance in Practice of String Hashing Functions'. In: *Database Systems for Advanced Applications '97*, pp. 215–223. DOI: 10.1142/9789812819536_0023. eprint: https://www.worldscientific.com/doi/pdf/10.1142/9789812819536_0023. URL: https://www.worldscientific.com/doi/abs/10.1142/9789812819536_0023.

[14] Tengfei Chang, Mališa Vučinić, Xavier V. Guillén et al. '6TiSCH Minimal Scheduling Function: Performance Evaluation'. In: *Internet Technology Letters* 3 (June 2020). Visited on 12th March 2021. DOI: 10.1002/itl2.170.

[15] Manas Khatua Karnish and Venkatesh Tamarapalli. 'IMSF: Improved Minimal Scheduling Function for Link Scheduling in 6TiSCH Networks'. In: *Proceedings of the International Conference on Distributed Computing and Networking (ICDCN)*. January 2022. DOI: 10.1145/3491003.3491027.

[16] David Hauweele, Remous-Aris Koutsiamanis, Bruno Quoitin et al. 'Pushing 6TiSCH Minimal Scheduling Function (MSF) to the Limits'. In: *HAL (Le Centre pour la Communication Scientifique Directe)* (July 2020). Visited on 16th October 2023. DOI: 10.1109/iscc50000.2020.9219692.

[17] Esteban Municio, Glenn Daneels, Marijana Vučinić et al. 'Simulating 6TiSCH networks'. In: *Transactions on Emerging Telecommunications Technologies* 30 (March 2019). Visited on 23rd April 2023, e3494–e3494. DOI: 10.1002/ett.3494.

[18] Francesca Righetti, Carlo Vallati and Sajal K. Das et al. 'An Experimental Evaluation of the 6top Protocol for Industrial IoT Applications'. In: *2019 IEEE Symposium on Computers and Communications (ISCC)*. 2019, pp. 1–6. DOI: 10.1109/ISCC47284.2019.8969590.

[19] Maria Rita Palattella, Thomas Watteyne, Qin Wang et al. 'On-the-Fly Bandwidth Reservation for 6TiSCH Wireless Industrial Networks'. In: *IEEE Sensors Journal* 16.2 (2016), pp. 550–560. DOI: 10.1109/JSEN.2015.2480886.

[20] Xavier Vilajosana et al. Marc Domingo-Prieto Tengfei Chang. 'Distributed PID-Based Scheduling for 6TiSCH Networks'. In: *IEEE Communications Letters* 20.5 (2016), pp. 1006–1009. DOI: 10.1109/lcomm.2016.2546880.

[21] Esteban Municio and Steven Latré. 'Decentralized broadcast-based scheduling for dense multi-hop TSCH networks'. In: *Proceedings of the Workshop on Mobility in the Evolving Internet Architecture*. MobiArch '16. New York City, New York: Association for Computing Machinery, 2016, pp. 19–24. ISBN: 9781450342575. DOI: 10.1145/2980137.2980143. URL: https://doi.org/10.1145/2980137.2980143.

[22] Nicola Accettura, Elvis Vogli, Maria Rita Palattella et al. 'Decentralized Traffic Aware Scheduling in 6TiSCH Networks: Design and Experimental Evaluation'. In: *IEEE Internet of Things Journal* 2.6 (2015), pp. 455–470. DOI: 10.1109/JIOT.2015.2476915.

[23] M. Palattella, N. Accettura, M. Dohler et al. 'Traffic aware scheduling algorithm for reliable low-power multi-hop IEEE 802.15.4e networks'. In: *Proc. IEEE 23rd Int. Symp. Pers. Indoor Mobile Radio Commun. (PIMRC)*. September 2012, pp. 327–332.

[24] T. Watteyne et al. 'OpenWSN: A standards-based low-power wireless development environment'. In: *Trans. Emerg. Telecommun. Technol.* 23.5 (2012), p. 48093.

[25] Yevhenii Shudrenko and Andreas Timm-Giel. 'Modeling end-to-end delays in TSCH wireless sensor networks using queuing theory and combinatorics'. In: *Computing* 106.9 (2024), pp. 2923–2947. ISSN: 1436-5057. DOI: 10.1007/s00607-024-01313-x. URL: https://doi.org/10.1007/s00607-024-01313-x.

List of Acronyms

IoT Internet of Things

WSN Wireless Sensor Network

MAC Medium Access Control

TDM Time Division Multiplexing

FDM Frequency Division Multiplexing

MTU Maximum Transmission Unit

HC Header Compression

6TiSCH IPv6 over the Time-Slotted Channel Hopping mode of IEEE 802.15.4e

6LoWPAN IPv6 over Low-Power Wireless Personal Area Networks

6top 6TiSCH Operation Sublayer

6P 6top Protocol

MSF Minimal Scheduling Function

SF Scheduling Function

KPI Key Performance Indicator

SAX Symbolic Aggregate approXimation

SoC System on Chip

RPL Routing Protocol for Low-Power and Lossy Networks

DAO Destination Advertisement Object

DIO DODAG Information Object

DODAG Destination-Oriented Directed Acyclic Graph

PDR Packet Delivery Ratio

UDP User Datagram Protocol

EUI-64 Extended Unique Identifier 64-bit

TSCH Time-Slotted Channel Hopping

ASN Absolute Slot Number

LLN Low Power and Lossy Network

CSMA/CA Carrier-sense Multiple Access with Collision Avoidance

EB Enhanced Beacon

SoC System on Chip