Thesis Dissertation

COMPARISON AND EVALUATION OF IEEE 802.15.4 (2015) TSCH AND GINSENG MAC

Antonios Stamatis

University of Cyprus

COMPUTER SCIENCE DEPARTMENT

May 2019

UNIVERSITY OF CYPRUS

COMPUTER SCIENCE DEPARTMENT

COMPARISON AND EVALUATION OF 802.15.4 (2015) TSCH AND GINSENG MAC

Antonios Stamatis

Supervisor
Dr. Vasos Vassiliou

A Thesis Submitted in Partial Fulfillment of the Requirements for the Degree of
Computer Science
at the University of Cyprus

May, 2019

ACKNOWLEDGEMENTS

With the current bachelor’s thesis, the long trip of my graduate studies comes to an end. At this point, I would like to thank everyone that contributed in their own way, to this achievement. Firstly, I would to thank my supervisor Dr. Vasos Vassiliou, for the support and guidance he provided from the start throughout the completion of this thesis. A big thanks, also goes to Dr. Christiana Ioannou and for the help that she provided in each step towards this completion. Last but not least, I would like to thank my family and friends, that have stood by me and supported me in any way and under any cost. Their faith in me was a catalyst for achieving my goals and dreams.

ABSTRACT

Due to the emergence of IoT and WSN techonologies there has been a significant ammount of research regarding their Data-link and routing layer desing. In this thesis, we examine Ginseng MAC and IEEE 802.15.4e (TSCH) which provide a possible way of node communication in a low powered and lossy network configuration. Taking a closer look at their implementation, and taking into account certain crucial metrics for a WSN, we run these configurations under various scenarios, providing the necessary results and come to a conclusion regarding their functionality and effectiveness in such network environments.

Contents

Chapter 1

Introduction

Contents

1.1 Motivation

Over the course of the last decade there have been many developments regarding wireless sensor networks and Internet of Things as a whole. We can see that with each passing day there is an increased interest regarding WSN in research, and ultimately, in our everyday lives. The most important aspect of a WSN is their ability to maintain an autonomous functionality. Thus the need for various protocols and standards regarding their usage have been in development and some of them are currently in use. One of the most important aspects is their mobility.

There have been many suggestions regarding a WSNs mobility scheme regarding the physical and data-link layer. Some could have better results than others regarding the topology, what type of information is send and whether the network is static or dynamic. Previous academic work *Ginseng MAC* has been researched while , also, there has been a technical standard released , *IEEE 802.15.4 (2015)* with its most recent version implementing *TSCH* [7].

1.2 Work Purpose

Examining these protocols under randomized traffic patterns provides a necessary discussion. Due to the emergence of IoT, having similar traits with WSN's, these protocols' design could be easily considered in their implementaion in an IoT environment.

The main aim of this thesis project is to evaluate both ginseng MAC and the newly standardized 802.15.4 (2015) TSCH in similar scenarios and gather information that will show which protocol is best suited for which situation. Furthermore, both protocols have their own scheduling algorithm , Dynamic Topology Control and Orchestra respectively, which will also be compared.

1.3 Work Methodology

In the first phase of the study, an extensive theoritical study was necessary in order to acquire knowledge regarding the design of WSN's in general and specifically the implementation regarding GinMAC, TSCH and the various scheduling algorithms for each.

In the second phase, getting acquainted with Contiki-NG was critical. Providing all of the above MAC and scheduling protocols that are being tested while also having a network simulator, it served our needs perfectly. Experimenting with various basic simulations and working with Cooja in order to figure out how simulations run and the best possible way to get the results that are in our interest.

Finally, in the third phase, the creation of necessary scripts , written in Python, were created for the sake of analysing log outputs produced by Cooja runs and producing the equivalent graphs. After finding the topologies that would serve our purpose for the thesis, we ran them multiple times with the different MAC/scheduling combinations and came to our conclusion.

1.4 Thesis Structure

Chapter 2 presents how Ginseng MAC and TSCH work and what is their goals regarding the desing of a WSN.

Chapter 3 provides a deeper inspection of GinMAC and TSCH regarding their scheduling and routing algorithms. Specifically , DTC for GinMAC and RPL, Minimal and Orchestra for TSCH.

Chapter 4 shows in what way the simulations where ran. Showing the topologies that were included for our tests, different parameters inserted in each simulation while also explaining the metrics that are useful for our results.

Chapter 5 presents the various results that were obtained from our simulation runs and gives explanations regarding their behavior.

Finally, in *Chapter 6*, we draw a general conclusion regarding GinMAC's and TSCH's designs and which one is better suited for which situation. Also, ideas on future works are presented at the end.

Chapter 2

Background

Contents

Wireless sensor networks (WSN) see deployments in areas where network performance assurance is not necessary (eg. agriculture) [19] . However, there are important areas where these assurances are essential in the installation and maintenance of a WSN such as in the case of a plant automation network. There have been developments in the creation of a MAC protocol where its main focus lies in the establishment of networks where there is a need to meet certain application-specific performance targets.

2.1 Ginseng MAC

Ginseng MAC (GinMAC) [17] is a sub-project of the EU-funded GINSENG project that aims to provide these needs in WSN. Specifically, it aims to provide assurances for reliability and delivery delay.

The main philosophy of GinMAC is to ensure data is being transported within a given time frame. Specifically, data traveling from sensors to sink (Ds) and data travelling from the sink to every actuator (Da). Any data arriving later than the given time bound is considered lost due to the time restraints and the importance of when data was sensed. Thus, a TDMA-based Medium Access Control (MAC) is considered for this approach.

The following features were implemented in GinMAC for the fulfillness of the WSN requirements:

Off-line Dimensioning: Before network establishment, network topology, traffic patterns and channel characteristics are known. With the examination of the above facts and calculation with the network's demands, a TDMA schedule of frame length F is created that each node must follow.

TDMA Schedule: TDMA slots are used, with a fixed size that ensures data transmission to destination and corresponding acknowledgement to sender. Each slot is used from only one node. There are three slot types [16] :

Basic Slots: Are the key slots for the GinMAC network to work. These consists of the necessary number of slots required for traffic to flow from all nodes to the sink and then, if needed, to each actuator. Each leaf node requires only one slot to forward to its parent. Each parent requires one slot for each of its children and one slot for its own data in order to be forwarded to its parent until, eventually, all data arrives at the sink. If there are actuators present, then slots are allocated from the sink to each actuator.

Additional Slots: Are used in the case where there are lost data packets that were traveling through the basic slots.

Unused Slots: Are present in a GinMAC schedule where the time that basic and additional slots cover are less than the delay requirement that is allowed ($F<\min\{Ds,Da\}$). These slots are then implemented in the frame so that the frame length is equal to the delay bound. In these slots, a node turns its transceiver off, reducing energy consumption in the process.

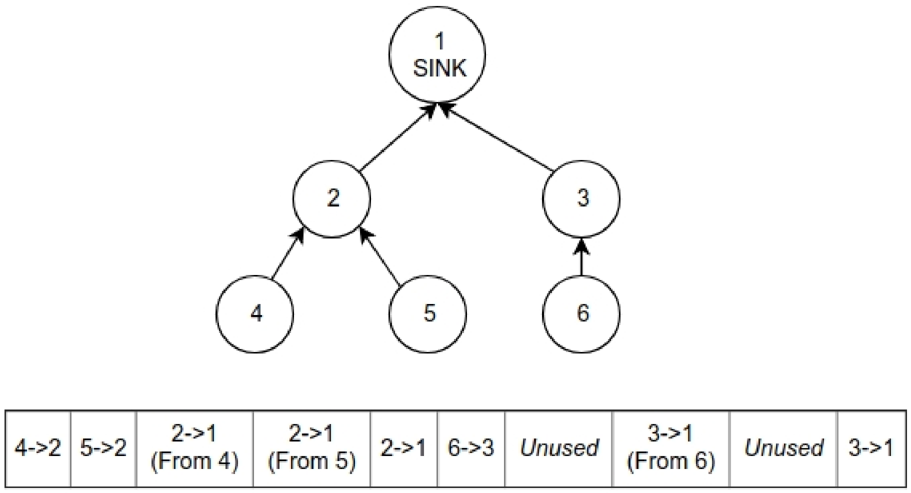


Figure 1: Simplified GinMAC TDMA Schedule

Reliability Control:When applying the above TDMA rules for each node, the protocol supports delays that the topology will require while providing high data transport reliability, due to the fact that each node has its unique slot time and with the inclusion of additional slots, expands the possibility of reliable transfer.

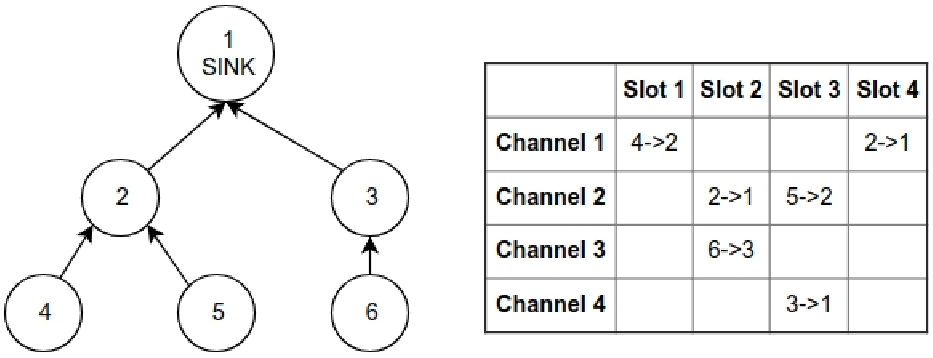
2.2 802.15.4 (2015) TSCH

The IEEE 802.15.4 is a technical standard that specifies the physical and data-link layer for low-rate wireless personal area networks [11]. Specified in the 2015 version is Time Slotted Channel Hoping (TSCH) which, at first glance, seems like a mixture of Time Division Multiple Access (TDMA) with Frequency-Division Multiple Access (FDMA) [9].

TSCH's logic is to build a globally synchronized mesh network. It divides time in slotframes. Each slotframe contains timeslots, around 10ms long, and for each timeslot there is a frame transmission and its acknowledgement. A timeslot can be distinguished by its time and channel offset. There is not a strict rule that forces slots to correspond to only one node, unlike GinMAC.

As its name suggests, TSCH uses channel hopping.[7] This means that for each slot there is a different transmission frequency for subsequent iterations. This minimizes the errors found from interference or multi-path fading because when a frame is lost due to a weak channel, it will automatically move to another in the next slotframe round.

The term channel means the sub-frequency band that will be used for a Tx and Rx at the specified time. Spefically, channel selection happens as follow: frequency = F{(ASN + channelOffset mod nFreq} [12], where F consists of a lookup table for all of the available channels which nFreq is the size of the table. ASN is a five-byte number that increments throughout the duration of a network's lifetime.



	Slot 1	Slot 2	Slot 3	Slot 4
Channel 1	4->2			2->1
Channel 2		2->1	5->2	
Channel 3		6->3		
Channel 4			3->1	

Figure 2: Simplified TSCH Schedule

In order for a node to join a TSCH network, it must listen for incoming Enhanced Beacon (EB) packets. These packets are broadcast at a fixed time by each node in order to allow further nodes to join the network.

Because of its reliance on scheduling, time synchronization is a key factor in the sustainability of the network. Time synchronization flows from the sink towards the leaf nodes. This sync between nodes is achieved whenever a node receives data or an ACK from its time source parent.

Chapter 3

Related Work

Contents

Further work was introduced in the aforementioned protocols in order to expand their capabilities.

3.1 Topology Discovery

3.1.1 RPL

The Routing Protocol for Low powered and lossy networks (RPL) [14] presents the currently available routing method in the 802.15.4 standard. It builds a Destination-Oriented

Directed Acyclic Graph (DODAG) where the root of the graph is the sink. This also enables the sink to have access to the internet.

When a sink decides to create an RPL network, it broadcasts packets called DAG Information Objects (DIO). Any node that wants to join the network respond to the DIO and , after being assigned a rank, they enter the topology. The rank is used to prevent loops and distinguish nodes from each other, parents children or siblings. This periodic DIO broadcast and response is passed to each child in order to provide the ability for more nodes to join. Also, it is used to refresh the DODAG in the case of any changes made to the physical topology. [18]

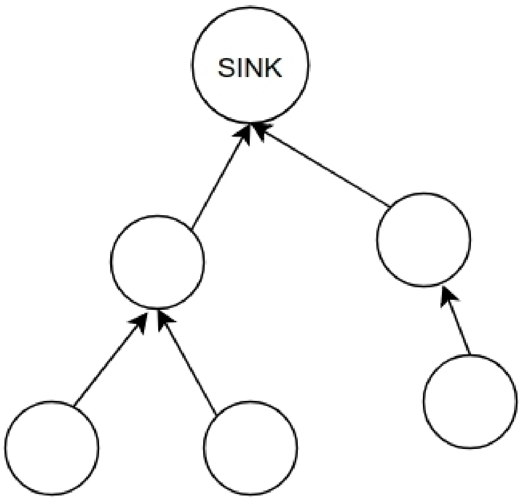


Figure 3: A DODAG instance

There are two implementations that are being used in Contiki-NG. The first is *RPL Classic* where there are multiple instances of DODAG's that are supported.
Then there is *RPL Lite* where there is only one DODAG instance provided in the network. This is preferred in the contiki implementation due to better performance and smaller ROM footprint [2].

3.1.2 DTC

Dynamic Topology Control (DTC) [15] is the proposed topology creation protocol that works in favor of GinMAC. It enables the creation of the GinMAC frame automatically without the need of a central entity or pre-existing knowledge of the network's coordinates.

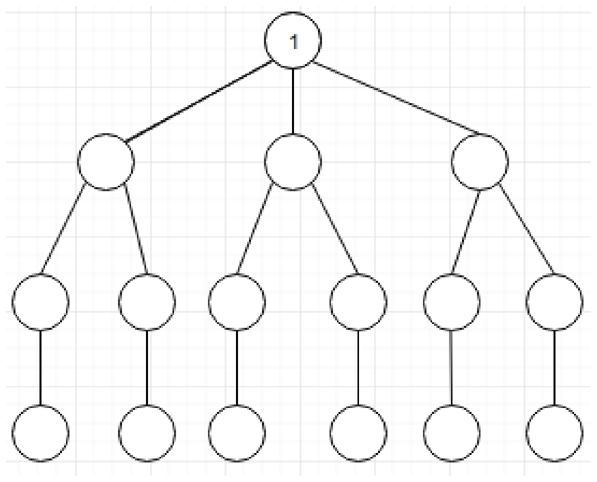


Figure 4: 3-2-1 Tree Formation

In contrast to RPL, it creates a tree topology for the network's needs. The tree formation is defined before implementation. The figure above shows a 3-2-1 tree topology which is the one being used in our evaluations. There are three control messages in DTC. *ADVERT* is used to promote the GinMAC network, the currently available children tree positions of the node that is sending the message and the positions available in the GinMAC frame. This is done by a node that is currently in the network. When a node retrieves an ADVERT message and wants to join, it sends a *JOIN* message, choosing randomly from the available tree positions available. When the JOIN message is then received by the original node, it sends a *JOIN ACK* message to confirm it's placement.

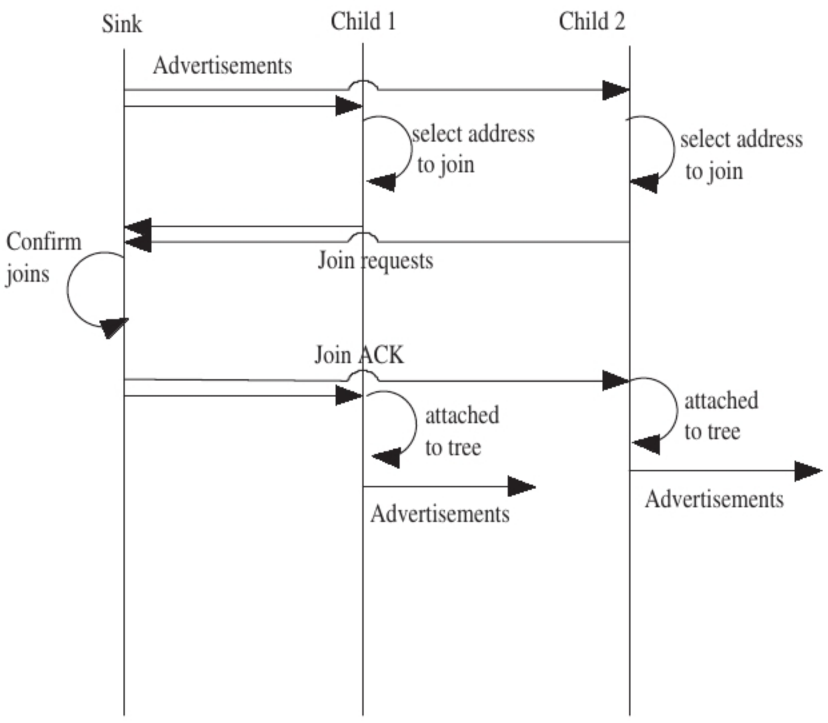


Figure 5: Message Sequence Diagram of Topology Control Mechanism [15]

This discovery process takes place in the beginning of the GinMAC frame, for upstream slots (for JOIN messages), and at the end (for ADVERT and JOIN ACK messages). The time duration of these upstream and downstream frames is called *epoch*. The duration is dependent on the chosen tree structure.

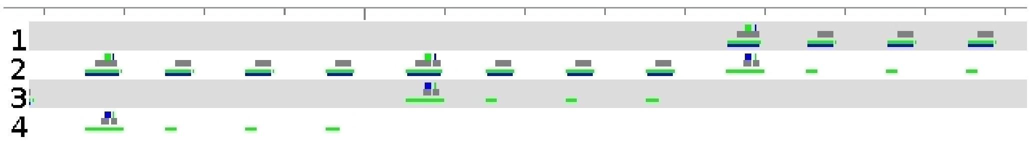


Figure 6: GinMAC TDMA Schedule

3.2 TSCH Scheduling

Scheduling for TSCH networks is a currently discussed topic around the network community. When in the case of a known and steady traffic pattern, a TSCH network can be build using a dedicated schedule pre-defined manually by a network administrator, commonly known as *TSCH Dedicated*.

When in the case of a randomly established network topology and traffic patterns there is the need of a more sophisticated and/or autonomous approach in the design of the schedule. We will explain two of these scheduling algorithms using the topology below where *node 1* is the sink.

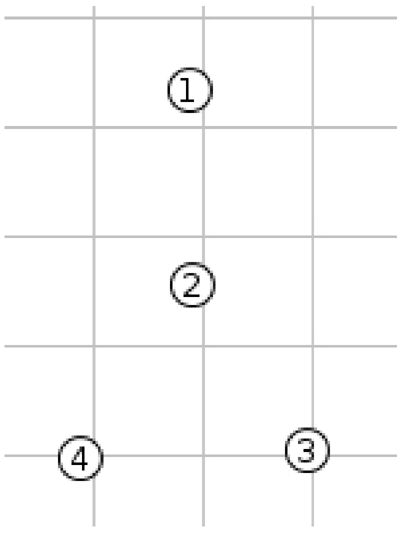


Figure 7: Simple 4 node network

3.2.1 TSCH Minimal

TSCH Minimal [8] states that there is a single shared slotframe for every node. This means that every node transmits/receives at the same time whenever there is data to be sent/received, even for broadcasting of EB packets.

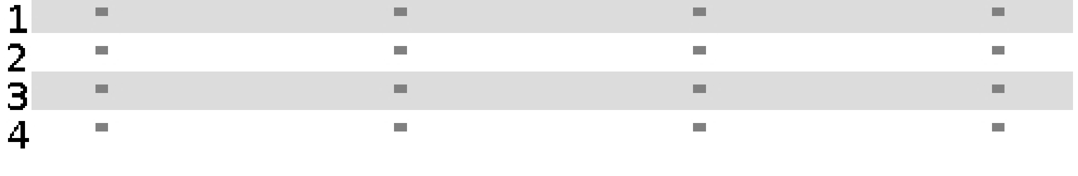


Figure 8: Minimal Schedule

This configuration provides the most basic form of a TSCH schedule because of the lack of any complex scheduling rules. As a result, there is a lack of consideration for the network topology and/or traffic flow.

3.2.2 Orchestra

Orchestra [10] is a scheduling algorithm that is the one proposed in the Contiki-NG implementation [1]. It is a decentralized autonomous scheduler. It uses the information created by RPL and TSCH and uses them to form it's schedule.

For TSCH time synchronization, a node uses it's RPL parent as it's preferred time source neighbor. This means that, whenever the topology changes, TSCH checks for any changes in the RPL structure and chooses it's new parent accordingly. Also, time sync happens whenever EB packets or RPL beacons are received by each node.

Orchestra has it's own method of handling slots. Each slot type is dedicated to a certain type of traffic.

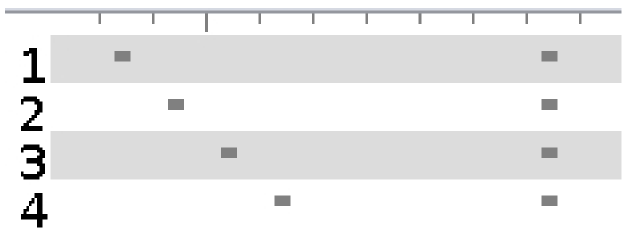


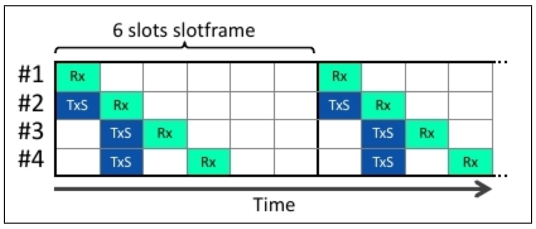
Figure 9: Sample Orchestra Schedule in Cooja (RBS/SBS Left, CS Right)

Common Shared Orchestra Slots (CS) are used exclusively by RPL in order to maintain the network topology, signals between parents and children while also allowing for RPL discovery by other nodes. This is where the TSCH schedules are maintained using information from the routing layer allowing for the creation of the following slot types.

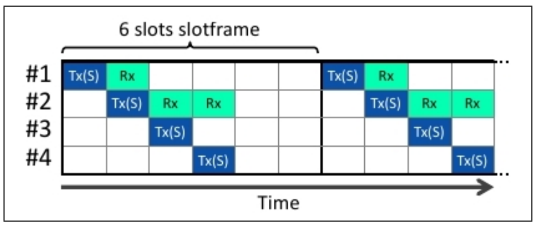
Receiver-based Shared Orchestra Slots (RBS) are used for data traffic and TSCH configuration data. Each node has a dedicated slot for Rx, based on the node's unique ID.

Thus every node's children knows when it's parent is ready for Rx and when it has something to send, then it has a Tx slot at the appropriate time as the parent's Rx. For every Rx slot, we have the appropriate Tx slots at any given time. Also, the node's unique ID functions as the channel offset for the channel selection.

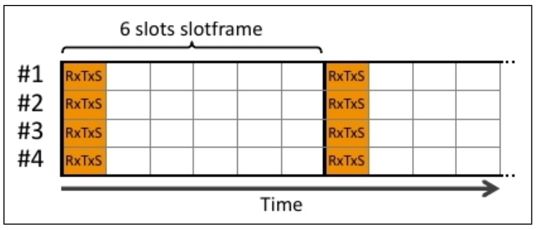
Sender-based Shared Orchestra Slots (SBS) follows the logic of RBS but in a reversed order. Instead of dedicated Rx slots, we have Tx slots based on the node's ID. Thus, for every Tx slot we have the analogous Rx slot, when a child wants to send data (Tx) the parent node must be ready to receive (Rx).



(a) Receiver-based Shared (RBS)



(b) Sender-based Shared (SBS)



(c) Common Shared (CS)

Figure 10: Sample Orchestra Slots [10]

3.3 CSMA/CA

Carrier Sense Multiple Access (CSMA) is a random access MAC protocol. It requires an always-on behavior by every node in the network. This enables the Rx at any given time and , whenever there is data to be sent, a Tx ”slot” is immediatly executed. Contiki-ng uses the Collision Avoidance (CSMA/CA) variation where a node listens before Tx in order to find whether there is any other traffic currently transmitting in the network/air. Despite this, an interference can happen in the case where two or more nodes send at the same time thus each signal interfers with the other. In this case, a random binary exponential backoff takes place in each node in order to re-send their data at a different time period [9].

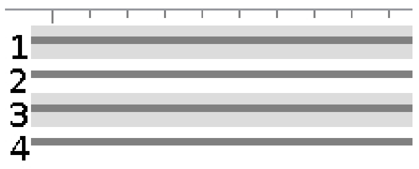


Figure 11: CSMA behavior in Cooja (Always-On)

This approach, logically, provides the fastest end-to-end delay. However ,due to CSMA's behavior of an always-on link, it is not at all a practical approach in the use of a WSN. More on that during scenario evaluations.

Chapter 4

Evaluation

Contents

4.1 Methodology

The focus of this paper is to compare these different types of link-layer, routing and scheduling protocols under different scenarios. Specifically, there are four different types of different network stack combinations that were used:

- Ginseng MAC (GinMAC) with Dynamic Topology Control (DTC)
- RPL + TSCH with Orchestra
- RPL + TSCH with Minimal Configuration

- RPL + CSMA/CA

These scenarios comprise of static topologies but with random data traffic patterns. Both of the scenarios comprise of 16 nodes , where *node 1* is the sink. Several tests were run with each scenario.

4.1.1 Contiki-NG

Contiki Next Generation (Contiki-NG) is an operating system (OS) that it's aim is towards IoT devices that are resource-contstrained. The NG variant enables IPv6 communication as well as full intergration of TSCH.

4.1.2 Cooja

All of the tests were conducted in a simulated environment using Cooja [3], which is the Contiki network simulator aimed for low powered networking devices. It's use was made under Contiki-NG in order to have access to IPv6 routing, TSCH and Orchestra features, since older version did not include one or the other.

For the testing of GinMAC, the use of virtual Sky motes were used for the simulations since it was originally built for those devices in mind. However, for the rest of the combinations, Cooja Motes, which are the basic virtual motes offered by Cooja, were used. This was due to the fact that TSCH with or without Orchestra could not load on a Sky mote because of their limited ROM size. This does not harm our tests though. Our interest in this paper is not about testing them under heavy traffic sizes.

4.2 Scenarios

For our tests, a real world topology was used and a random topology created solely through Cooja.

4.2.1 Ginseng Refinery Test Bed

This was a real life refinery installation of a WSN to measure performance of GinMAC and DTC [13].

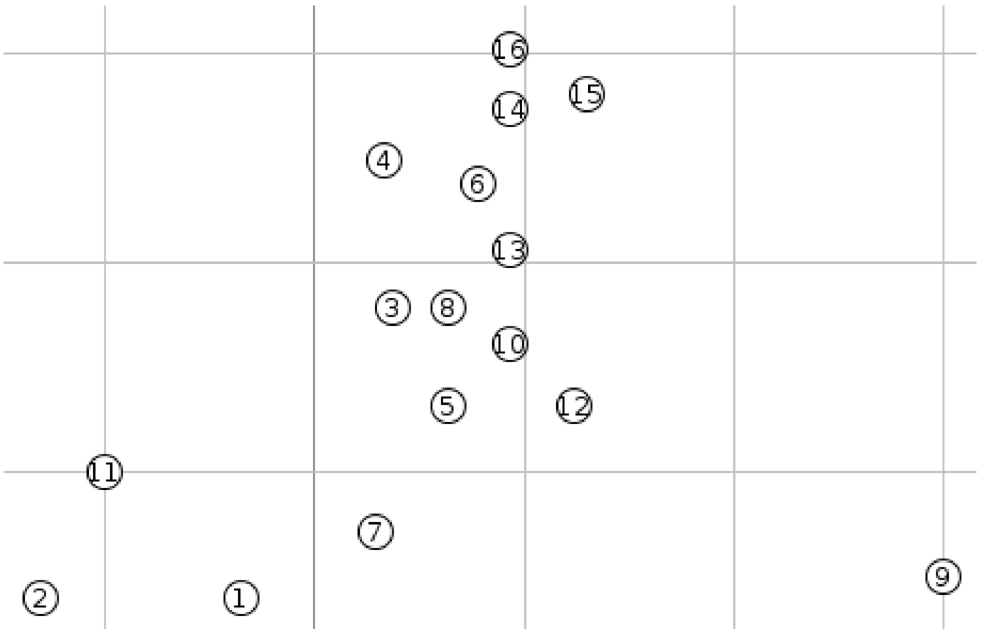


Figure 12: Refinery Scenario Topology

Every node is in the range of communication with every other node, thus enabling all network combinations to create their DODAG or tree without any distance limitations.

4.2.2 Random Test

This is a randomly generated topology created exclusively in Cooja, in a 100m x 100m Square grid.

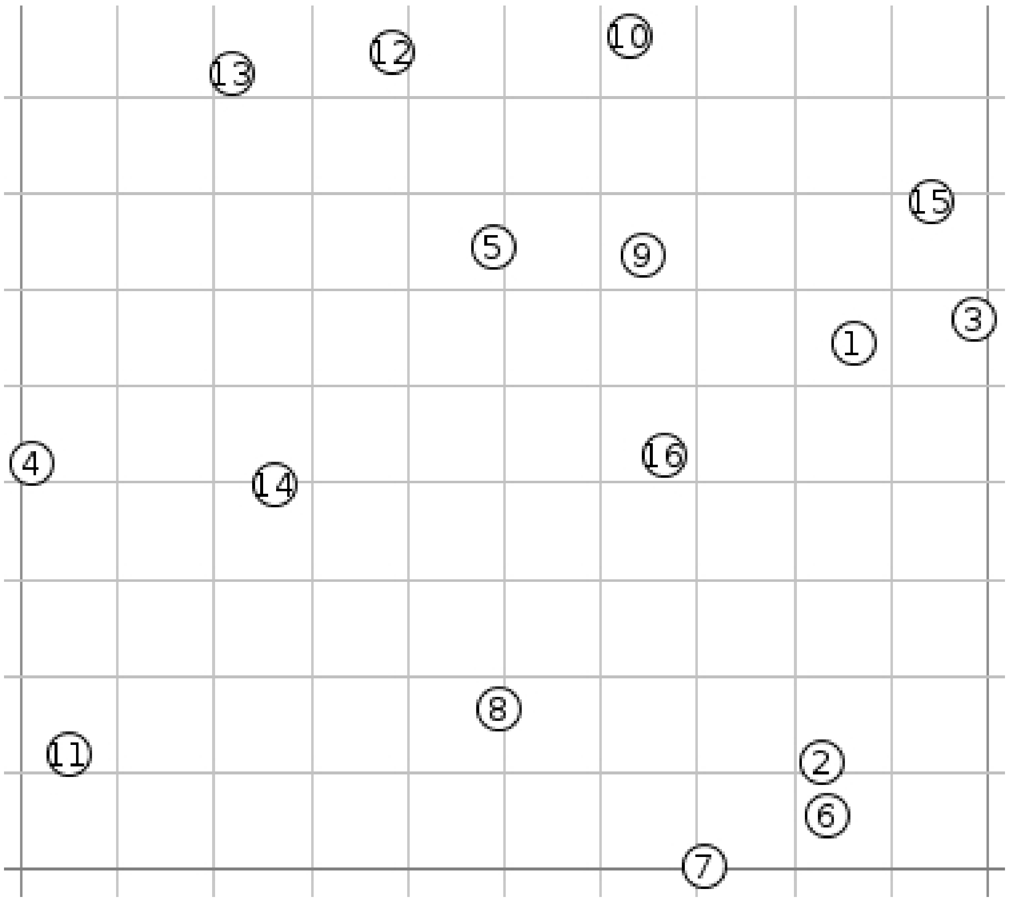


Figure 13: Random Scenario Topology

The difference here is that not all nodes can communicate directly (logically) with another.

4.3 Simulations

4.3.1 Parameters

We ran each diffrent combination of MAC protocols, with random traffic patterns. Specifically, each node generates packets at a given time. Our times range between one to ten seconds. This means in the one second scenario every node generates packets to send every one second, in the two second every two and so on. We apply these parameters to each combination, measure the log outputs from each test and generate various graphs showcasing the network's behavior.

Slotframes size for each combination are:

- **Orchestra**: 10ms for CS. 160ms for RBS. There are 16 nodes in the network, and each has its own receiver slot, based on the node's ID, with 10ms each.
- **Minimal**: 10ms. Every node operates at the same timeslot.
- **GinMAC**: 760ms for TDMA frame.
- **CSMA/CA**: No slotframes of any kind. Always-on behavior.

4.3.2 Evaluation Metrics

Our metrics that we will examine are as follows:

- *Tree Structure*: Examine the shape of complete tree structure (aka. all nodes have a path to the sink), depth of the tree and whether it's balanced or not. Basically compare RPL and DTC on how they decide where a node takes which place in the tree structure.
- *Tree Setup Time*: Check how long a tree needs to be constructed between RPL and DTC
- *Packet Delivery Rate (PDR)*: Depicts the percentage of packets that were sent towards the sink compared to the actual ammount of packets that were produced, even the ones that were not sent due to network errors.
- *Packet Reception Rate (PRR)*: Depicts the percentage of packets that arrived succesfully at the sink compared to the total number of packets sent.
- *Average Latency*: The Average end-to-end delay in the network.

- *Energy Consumption*: Measure the overall energy consumption. There is not a way to get actual energy consumption, due to most simulations including cooja nodes. Thus energy consumption is measured with the overall precentage of how long each node’s antenna is active compared to the overall test time.

Chapter 5

Results

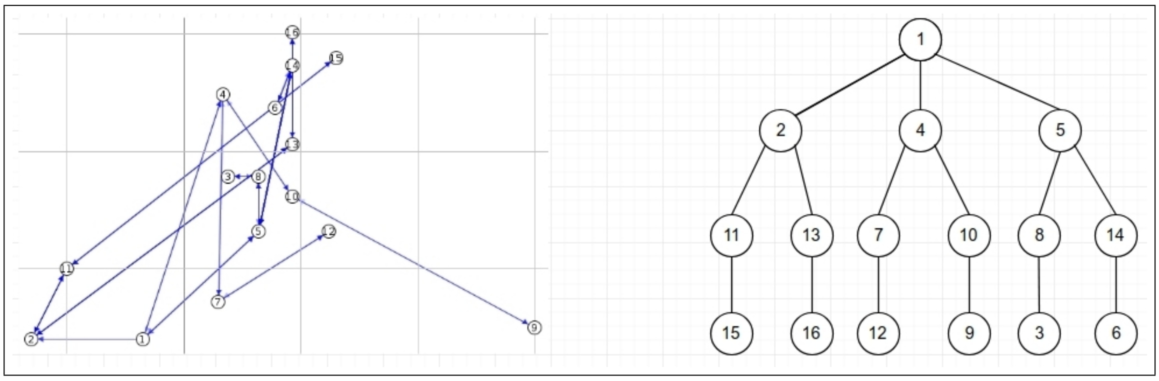
Contents

In this chapter, will be presenting the results that were acquired from our test runs, and compare between the different MAC/Scheduling combinations.

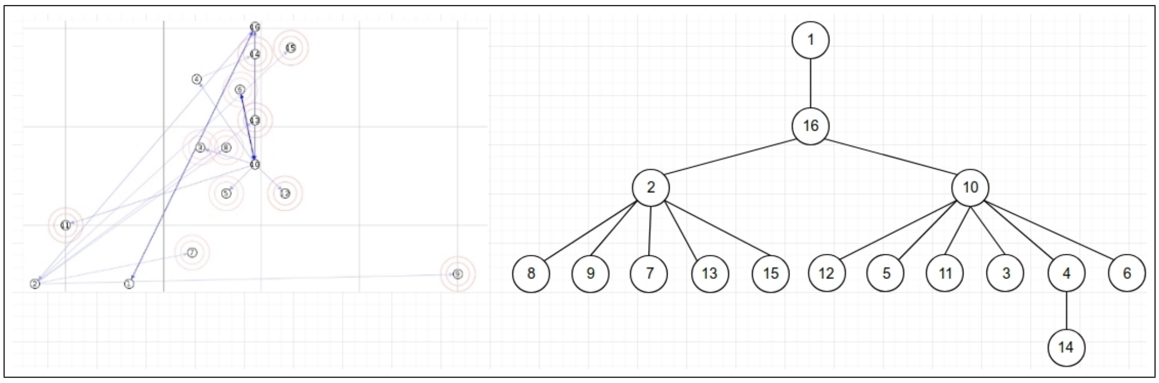
5.1 Refinery scenario tests results

5.1.1 Tree Structure

The way in which they handle their tree structure is quite different. The most obvious is that the way DTC builds the tree provides a much more balanced version than that of RPL.



(a) DTC



(b) RPL

Figure 14: Refinery Scenario Tree Formations

Because, RPL works in a much more randomized way, the resulting tree (DODAG) does not always provide the most balanced way to represent the network. The tree shown in the image is one of the many trees that RPL created in our simulations.

On the contrary, DTC works in a more restricted manner. Depending on the tree configuration before initialization, in our case a 3-2-1 tree, it not a randomized tree such as in the case of RPL. That's the reason that the final tree is balanced. Also, in every run, the same tree setup is present.

5.1.2 Tree Setup Time

It should be noted that these tests were recorded for tests that had five to ten second intervals. In faster intervals, Orchestra and Minimal did not manage to create the whole tree topology.

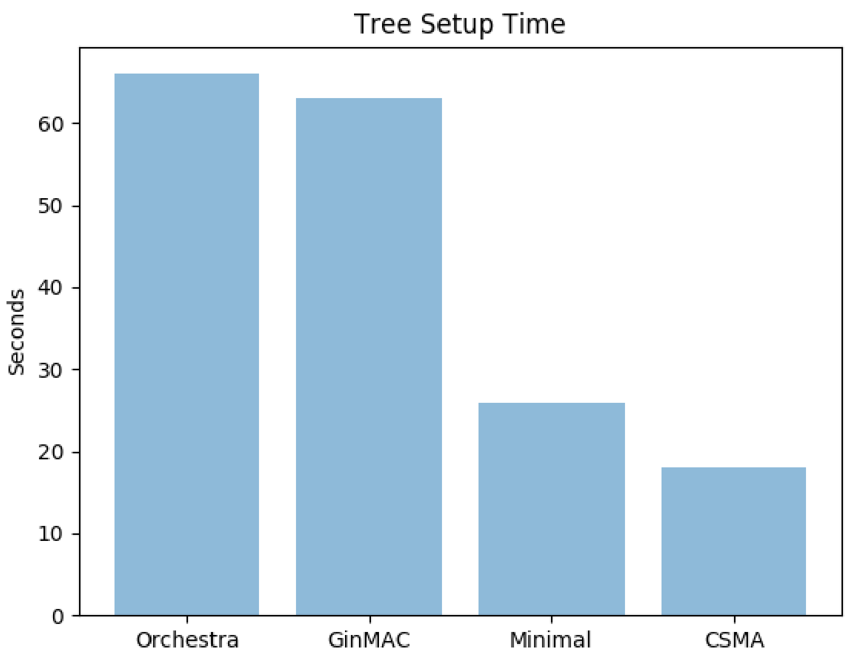


Figure 15: Tree Setup Time Refinery

The best case of RPL can be seen with CSMA. Because there is not any slot restriction, we can see how RPL works and completes it's tree setup in the least ammount of time, which in our case is just below 20 seconds.

Minimal shows a rather interesting result where it is actually the fastest of the other slotted type protocols in it's tree setup time. Slots are not as spreaded as much as Orchestra, thus giving RPL the time to create it's topology much faster.

Orchestra and GinMAC show us that their setup time is about the same for both. GinMAC however, managed to create it's tree topology even in the fastest interval times giving it a huge advantage compared to Orchestra or Minimal.

5.1.3 Packet Delivery

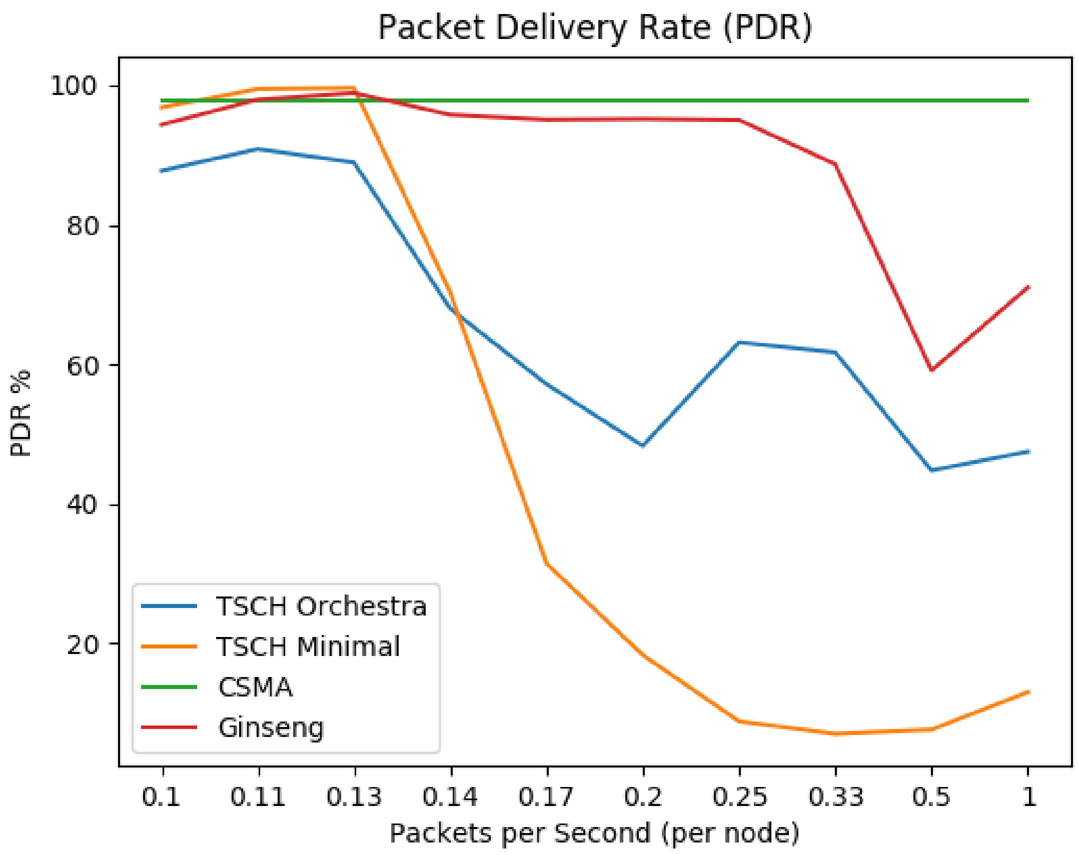


Figure 16: Packet Delivery Rate (PDR) Refinery

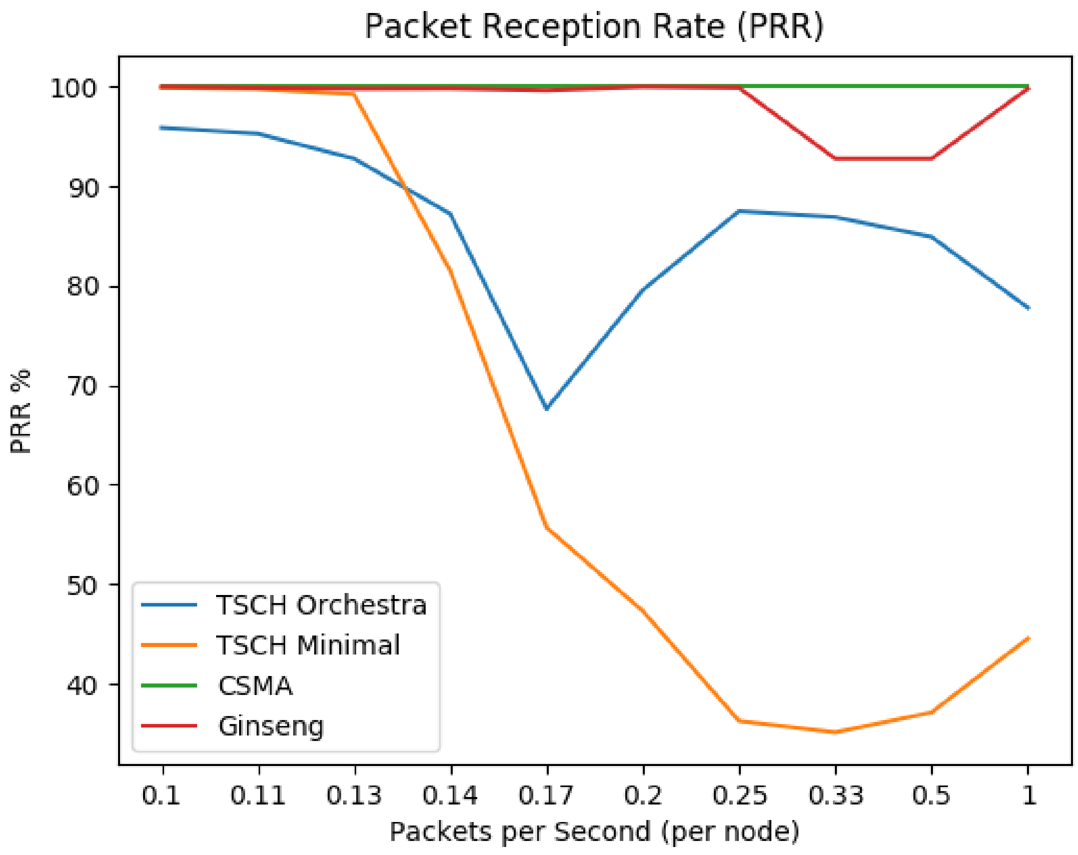


Figure 17: Packet Reception Rate (PRR) Refinery

On first glance of the following graphs we can come to the conclusion that the best overall protocol in terms of PDR, PRR and average end-to-end delay happens to be CSMA/CA. Despite this, it is actually the worst choice for a WSN because there is not any philosophy regarding power consumption. It maintains an always-on link and our nodes would run out of power fairly quickly. It is present in our graphs only for comparison reasons.

In more congested traffic patterns, the worst performance out of all happens to be TSCH Minimal. Due to this high rate of traffic, packet's are congested in the ”schedule” that is implemented. This single common shared slot for all nodes presents a very 'unhealthy' network where, in faster traffic times, most of the nodes in the network did not even manage to enter it. Even if the outer nodes of the tree managed to enter, usually they would, after some time, drop out of the network because of the long delays between sending and acknowledgemnts of RPL messages. At slower traffic sending intervals, Minimal behaves as well as CSMA in terms of PDR and PRR.

This RPL behavior is also present in the TSCH Orchestra runs, with somewhat better results. In faster traffic intervals, there is a greater connectivity in our network. However, for the same reasons as Minimal, some nodes do not even get a chance at sending data, because of the lack of routing knowledge for that node. Essentially, there is no path found from that node towards the sink. RPL is not getting enough time in order to build it's network topology (DODAG) and thus allowing TSCH to work.

GinMAC, on the other hand, doesn not have those limitations. It manages to build it's tree topology using DTC, fairly quickly, and most of the network's actions consist of the transport of useful data rather than topology information. That is why, there is not a single run in our scenario where GinMAC does not fully work with all nodes. In every test, GinMAC works with almost no packet losses. The only time where packets are indeed dropped is in its initialization where DTC is still trying to build the tree.

5.1.4 Latency

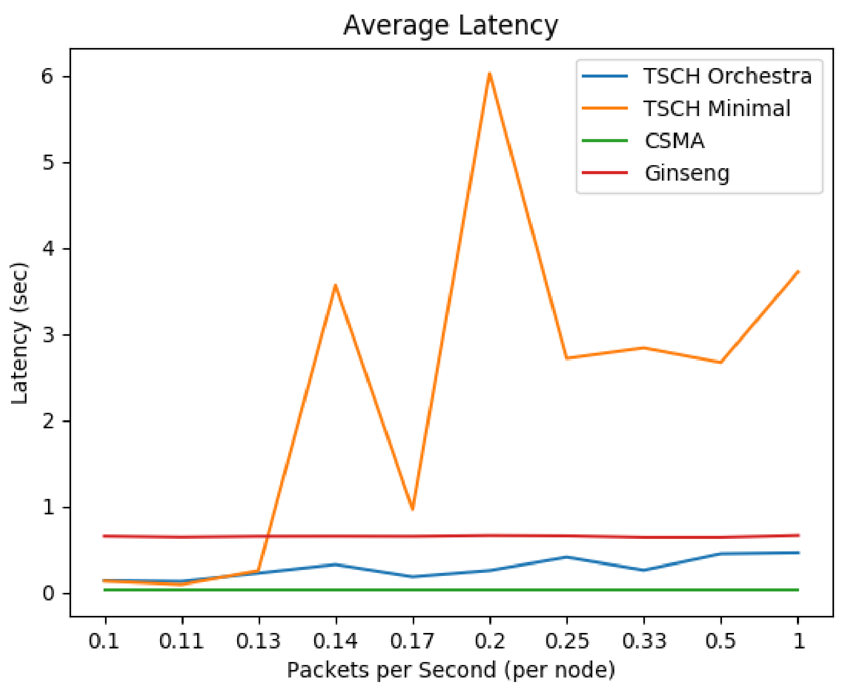


Figure 18: Average Latency - Refinery

The end-to-end delay of a network is especially crucial due to the nature of time contraints in a WSN. CSMA provides the best solution for minimum latency but, as stated above, it's the worst in the long run.

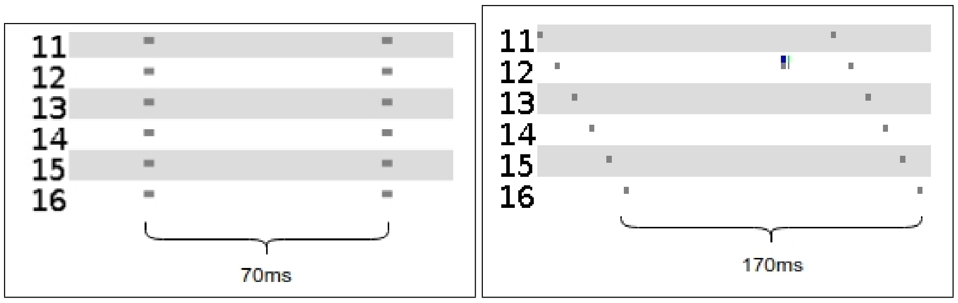
TSCH Minimal also seems to struggle in faster traffic intervals in terms of latency. Due to the single slot principle there is a high ammount of delay when multiple packets are being transmitted at the same time. This leads to many packet retransmissions due to co-channel interference and packets tend to wait a lot until their correpsonding time slot to be transmitted comes. When traffic intervals are at a slower pace, then the average latency drops at about the same as Orchestra's case.

Orchestra's latency fairs quite better compared to Minimal in faster traffic sending intervals. This happens because there is not a lot of congestion in the network when timeslots are divided between nodes, rather than co-existing at the same time. Latency is about the same for all runs, despite how much traffic is being send. The only case when this is not true is at the one Second interval, but the end-to-end delay is nowhere near as much as the case in Minimal. The average Latency accross all runs seems to be around 0.5 seconds. This means that , on average, a total of three slotframes (160ms size each) are needed for the transmission of a packet from its source towards the sink.

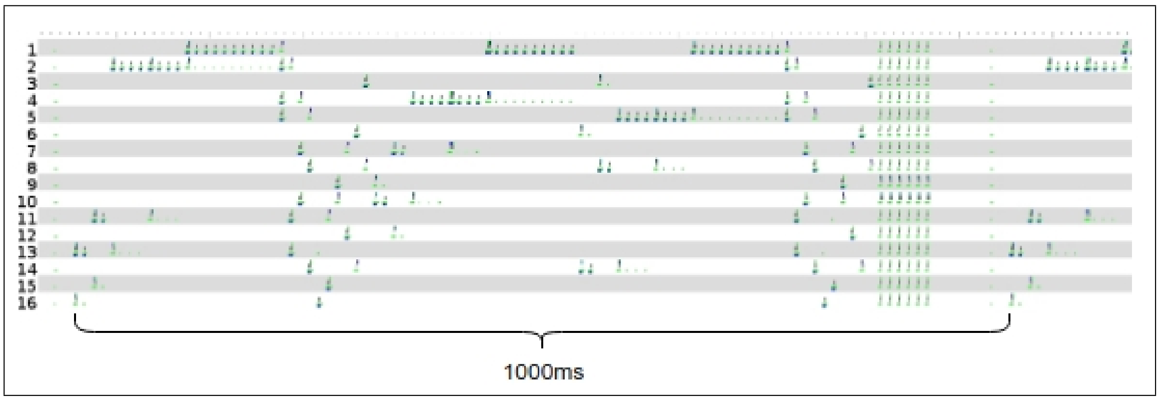
The most 'predictable' latency behavior is observed in GinMAC. The way that the TDMA schedule is constructed in order to comply with the Timed-Critical data delivery philosophy is the main reason for this behavior. Clocking at around 0.7-0.8 seconds, this average is also directly responsible by the way DTC creates the GinMAC tree.

TSCH runs provided us with overall better latency due to its not so restricted slot usage whereas GinMAC has a higher latency due to it's restricted slot formation.

Basically, the real issue that presents this high latency number in GinMAC is its time between subsequent slots for a node. Meaning, the amount of time a node has to wait in order to send again. In GinMAC this time reaches close to around one second. While in TSCH, this figure is lower: 70ms and 170ms in Minimal and Orchestra respectively. In Orchestra, when the use of RBS slots are used, as it is the case in our tests, then this time is even less depending on which node is the RPL parent of each node. The figure below shows this in the case of node 16.



(a) Minimal (b) Orchestra



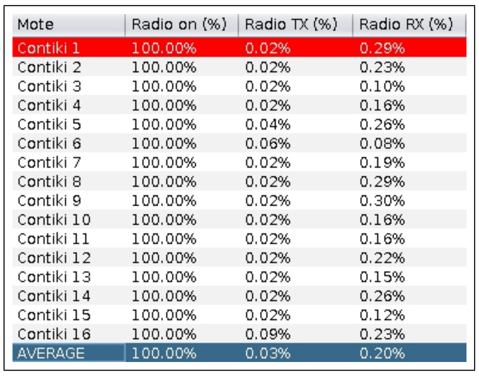
(c) GinMAC

Figure 19: Time between slots for a single node

5.1.5 Power Consumption

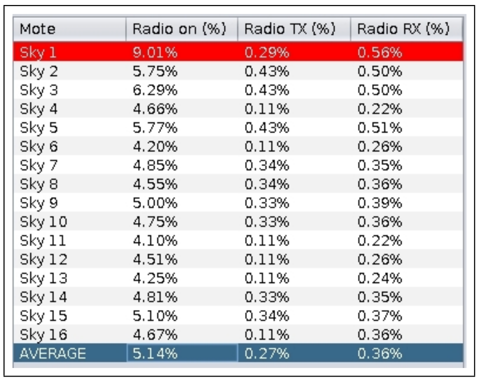
Power consumption is, for many, the most important aspect when designing a protocol for a WSN. Even when the delivery of data is succesful, when the power consumption is not reduced at a low number, then that WSN protocol is pretty much useless.

The figure below shows us the amount of power consumption for each of our MAC combinations. Specifically, the tables below show the percentage of time each node was active compared to the overall running time. The last row of each table give us an average of every node. These results were produced when we ran each combination for 30 minutes.



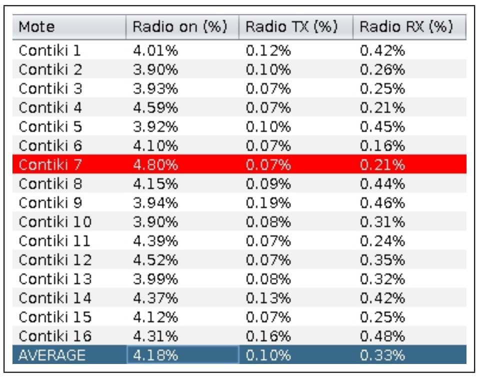
Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Contiki 1	100.00%	0.02%	0.29%
Contiki 2	100.00%	0.02%	0.23%
Contiki 3	100.00%	0.02%	0.10%
Contiki 4	100.00%	0.02%	0.16%
Contiki 5	100.00%	0.04%	0.26%
Contiki 6	100.00%	0.06%	0.08%
Contiki 7	100.00%	0.02%	0.19%
Contiki 8	100.00%	0.02%	0.29%
Contiki 9	100.00%	0.02%	0.30%
Contiki 10	100.00%	0.02%	0.16%
Contiki 11	100.00%	0.02%	0.16%
Contiki 12	100.00%	0.02%	0.22%
Contiki 13	100.00%	0.02%	0.15%
Contiki 14	100.00%	0.02%	0.26%
Contiki 15	100.00%	0.02%	0.12%
Contiki 16	100.00%	0.09%	0.23%
AVERAGE	100.00%	0.03%	0.20%

(a) CSMA



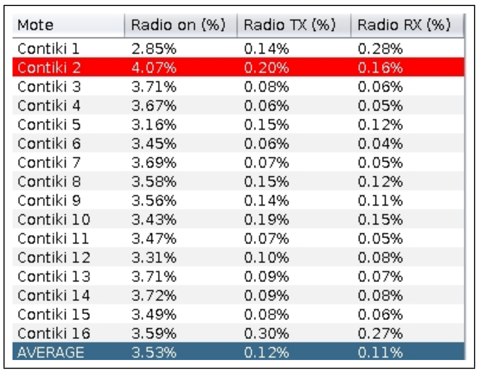
Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	9.01%	0.29%	0.56%
Sky 2	5.75%	0.43%	0.50%
Sky 3	6.29%	0.43%	0.50%
Sky 4	4.66%	0.11%	0.22%
Sky 5	5.77%	0.43%	0.51%
Sky 6	4.20%	0.11%	0.26%
Sky 7	4.85%	0.34%	0.35%
Sky 8	4.55%	0.34%	0.36%
Sky 9	5.00%	0.33%	0.39%
Sky 10	4.75%	0.33%	0.36%
Sky 11	4.10%	0.11%	0.22%
Sky 12	4.51%	0.11%	0.26%
Sky 13	4.25%	0.11%	0.24%
Sky 14	4.81%	0.33%	0.35%
Sky 15	5.10%	0.34%	0.37%
Sky 16	4.67%	0.11%	0.36%
AVERAGE	5.14%	0.27%	0.36%

(b) GinMAC



Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Contiki 1	4.01%	0.12%	0.42%
Contiki 2	3.90%	0.10%	0.26%
Contiki 3	3.93%	0.07%	0.25%
Contiki 4	4.59%	0.07%	0.21%
Contiki 5	3.92%	0.10%	0.45%
Contiki 6	4.10%	0.07%	0.16%
Contiki 7	4.80%	0.07%	0.21%
Contiki 8	4.15%	0.09%	0.44%
Contiki 9	3.94%	0.19%	0.46%
Contiki 10	3.90%	0.08%	0.31%
Contiki 11	4.39%	0.07%	0.24%
Contiki 12	4.52%	0.07%	0.35%
Contiki 13	3.99%	0.08%	0.32%
Contiki 14	4.37%	0.13%	0.42%
Contiki 15	4.12%	0.07%	0.25%
Contiki 16	4.31%	0.16%	0.48%
AVERAGE	4.18%	0.10%	0.33%

(c) TSCH Minimal



Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Contiki 1	2.85%	0.14%	0.28%
Contiki 2	4.07%	0.20%	0.16%
Contiki 3	3.71%	0.08%	0.06%
Contiki 4	3.67%	0.06%	0.05%
Contiki 5	3.16%	0.15%	0.12%
Contiki 6	3.45%	0.06%	0.04%
Contiki 7	3.69%	0.07%	0.05%
Contiki 8	3.58%	0.15%	0.12%
Contiki 9	3.56%	0.14%	0.11%
Contiki 10	3.43%	0.19%	0.15%
Contiki 11	3.47%	0.07%	0.05%
Contiki 12	3.31%	0.10%	0.08%
Contiki 13	3.71%	0.09%	0.07%
Contiki 14	3.72%	0.09%	0.08%
Contiki 15	3.49%	0.08%	0.06%
Contiki 16	3.59%	0.30%	0.27%
AVERAGE	3.53%	0.12%	0.11%

(d) TSCH Orchestra

Figure 20: Power Consumption (30 Minute Runs)

CSMA is the main ”victim” of this trope. An always-on link means a 100 percent antenna usage through the network’s lifecycle. Building a WSN with CSMA provides a reliable way of data transportation but with a very limited time of operation.

GinMAC has a higher amount of power consumption compared to TSCH related runs. A large amount of this is because of DTC where it basically requires more amount of antenna use in order to function.

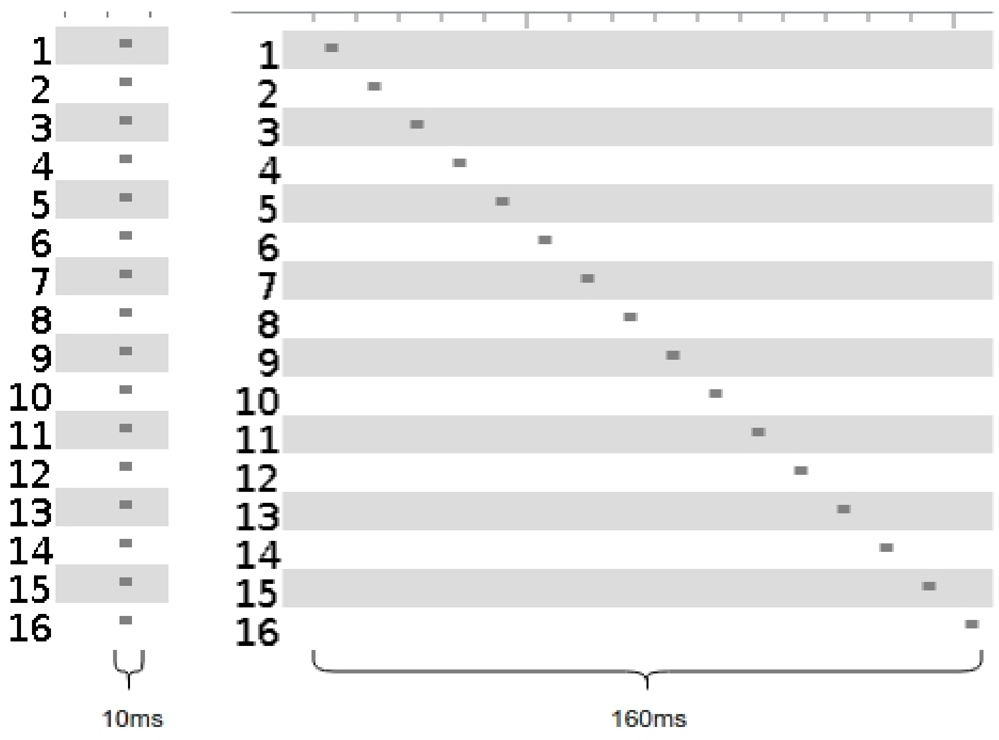


Figure 21: Slotframe Lengths (Minimal,Orchestra)

TSCH Minimal has a lower power consumption than GinMAC and Orchestra has even less. This is explained by the way orchestra speads the slots evenly in each node rather than Minimal. This means that a 10ms slotframe in Minimal , where every antenna is active, is spreaded in a 160ms slotframe.

5.2 Random scenario tests results

Our results for the second scenario are pretty much the same as the first. The only real difference we can see is due to the network's area. Our topology covers a larger ground. As a result, not every node's signal can reach every other node.

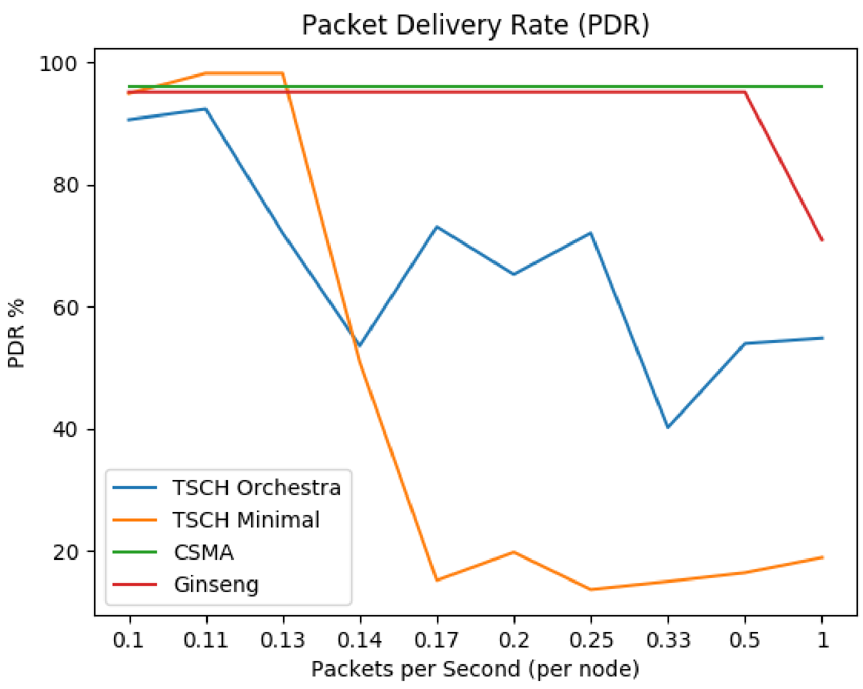


Figure 22: Packet Delivery Rate (PDR) Random Topology

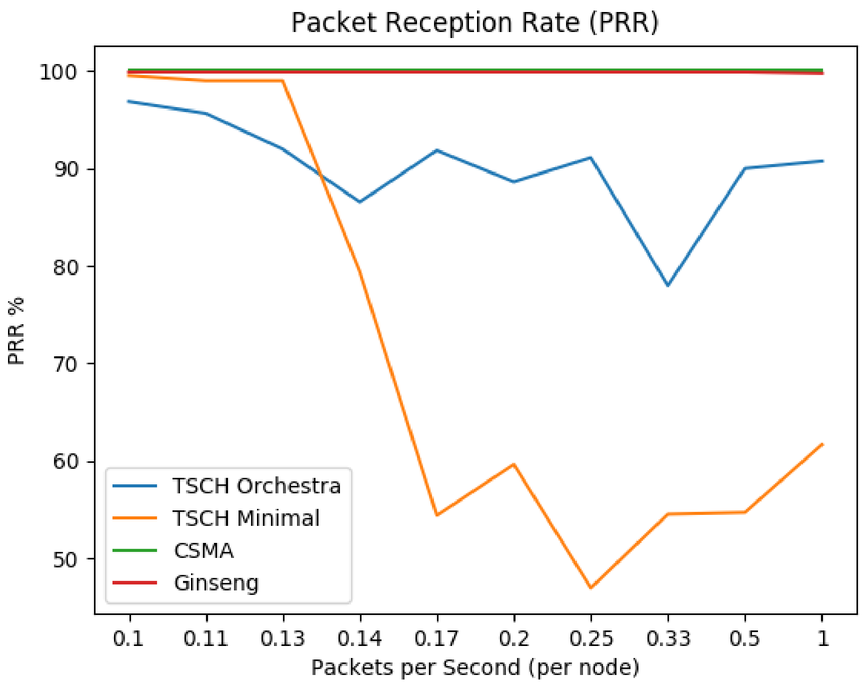


Figure 23: Packet Reception Rate (PRR) Random Topology

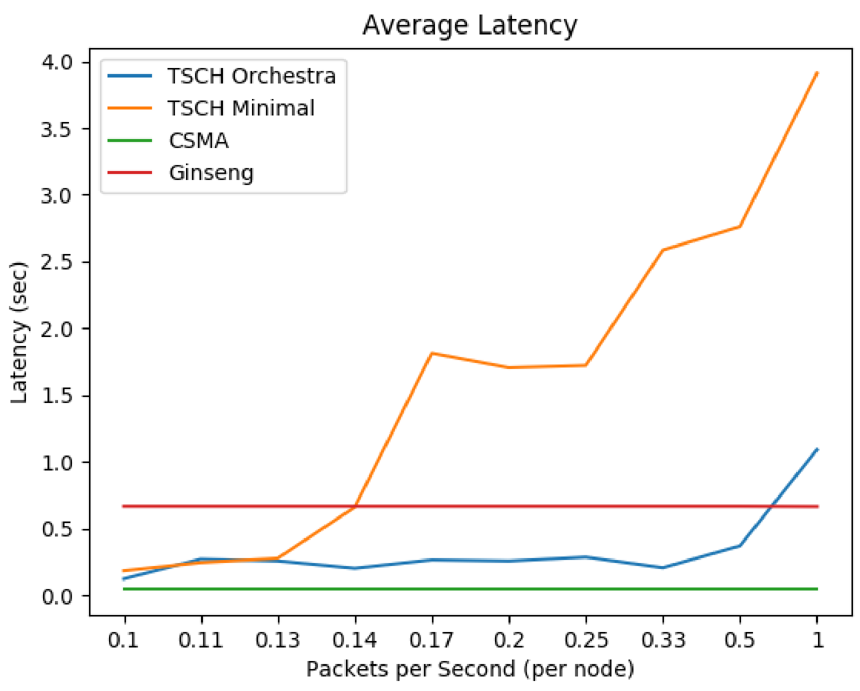
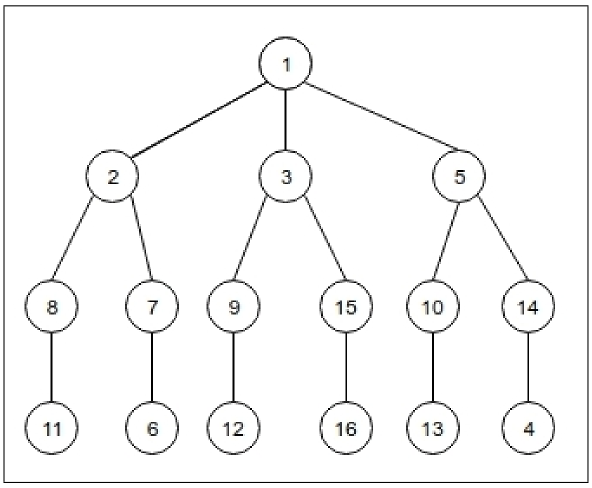


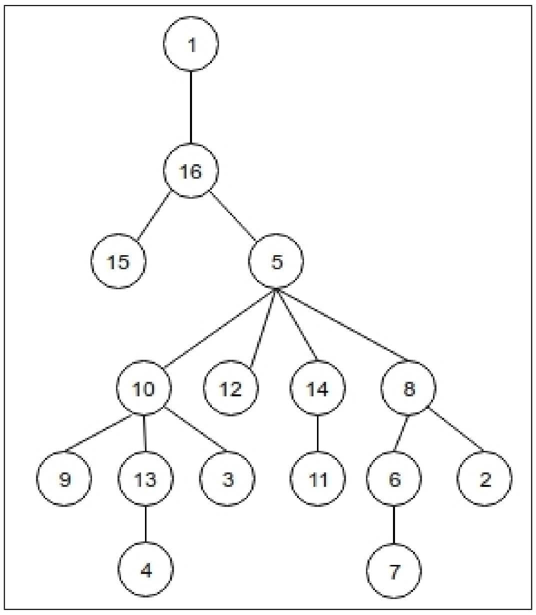
Figure 24: Average Latency - Random Topology

CSMA works flawlessly as expected. TSCH Minimal and Orchestra present their results as expected. In higher traffic send intervals, there is a higher latency while also a lower PDR and PRR for both. It should be noted that Minimal achieves almost 100 percent PRR in higher intervals while Orchestra does not in any test run. Average latency for Orchestra clocks at around 0.2 - 0.3 on average. This occurs in Minimal only in higher traffic intervals.

GinMAC works in a similar manner as in the previous scenario. PDR and PRR are at desired ammounts while keeping the average latency always the same, despite the frequency in traffic data. GinMAC's latency is also at a higher value than TSCH.



(a) DTC



(b) RPL

Figure 25: Random Scenario Tree Formations

The way that DTC and RPL handle their tree structure is reminiscent of the previous test. DTC , pre-defined, has a 3-2-1 tree structure while TSCH/RPL has a more versatile tree configuration.

5.3 Random Scenario with Node Deadlock

Following a slight variation in our Random Topology scenario, we have come accross a rather interesting result regarding GinMAC. Our difference is the following node position change (node 16):

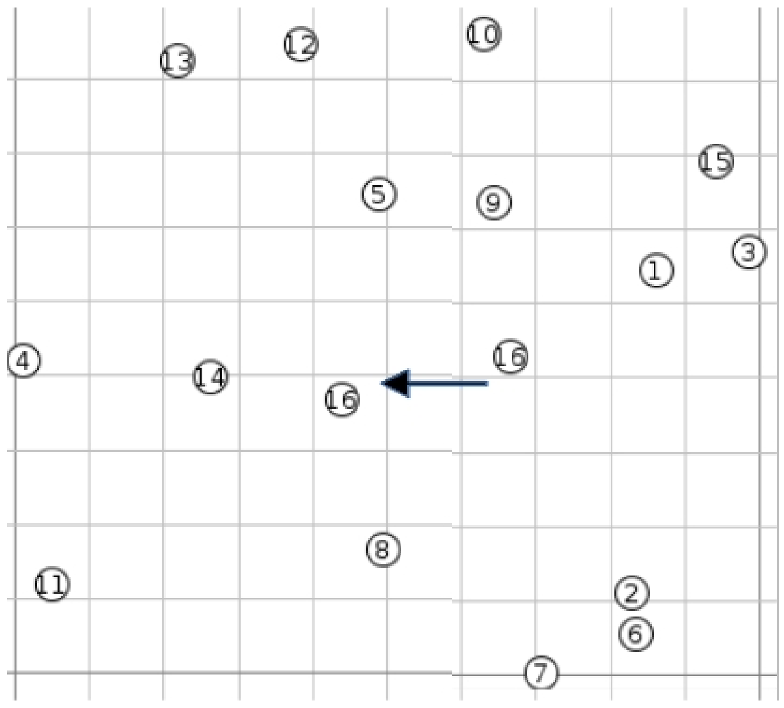
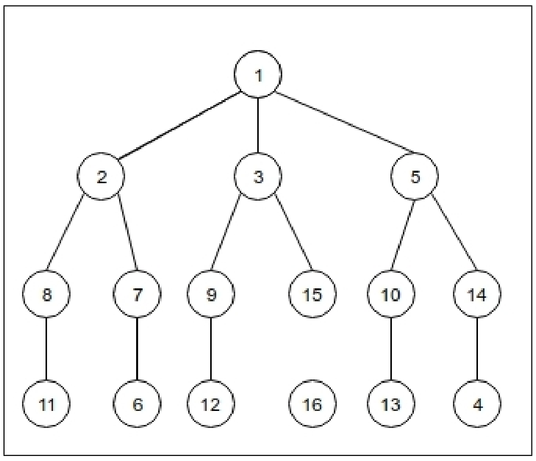


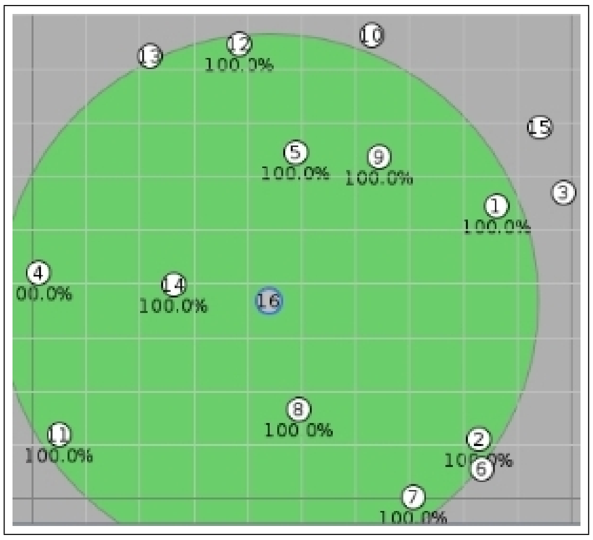
Figure 26: Random Scenario (Node 16 Position Change)

CSMA or TSCH do not show any kind of significant difference in our results (Same results as original Random Scenario). However, GinMAC did not work as expected. Specifically, *node 16* did not manage to enter the network in any of our runs. This is due to DTC's handling of the tree formation.

In each of our runs, *node 16* was one of the last nodes to decide to enter the GinMAC frame. However, when it tried to enter the frame, there was not any space available for it in the tree because every other possible one was already taken by another node. The only space left was the child of *node 15*, which is not in the vicinity of *node 16*. DTC's behavior does not involve the switch between nodes in the tree.



(a) Current Tree



(b) Node 16 Receiver Radius

Figure 27: Node blocking instance

This resulted in node 16 remaining outside the GinMAC frame indefinetely. Every other possible position in the frame/tree was occupied.

Chapter 6

Concluding Remarks

Contents

6.1 Conclusion

In this thesis we have tested various Data-link and routing protocols regarding the creation and maintanance of a WSN. One of the most critical metrics regarding the design of a WSN should be it's power consumption, due to the importance of a network's longevity. This is especially important in the WSN's installment in out-of-reach areas that a person's physical contact with the WSN is not possible. The other is the amount of useful data that are indeed collected by nodes and passed to the sink succesfully (PRR). Considering the above, and the results gathered in *Chapter 5*, we can come to a safe conclusion that the best combinations that satisfy those needs are GinMAC and TSCH Minimal.

GinMAC, because of its philosophy for Time-critical data delivery, provides the best overall PDR and PRR out of the rest. This can be seen in its slot design where it provides a guarantee in data delivery for every data collected from all nodes towards the sink in the way it handles basic slots and the inclusion of additional slots for backup slots. The only real issue with GinMAC is the way it handles its tree structure, specifically DTC. Despite doing what was intended, creating a tree topology that satisfies the network's needs, it actually constraints the way the tree is created due to the fact that a tree configuration must be considered before network initialization. This is necessary for the way in which slots (basic, additional, unused) are to be spread in each level of the tree topology.

TSCH variants showed interesting results compared to GinMAC. Because of its lack of timed data delivery, as in GinMAC, there is a serious lack in PRR for both TSCH variants, Minimal and Orchestra, in faster traffic intervals. However, in less congested network traffic, TSCH works as intended. In particular, Minimal provided us with a healthy network were PRR was almost at 100 percent, and its latency was almost as equal to GinMAC's. Providing TSCH with a more complex scheduler, Orchestra, it showed that it worked against TSCH and RPL. This can be attributed to its handling of RBS and SBS slots where a node's time to resend traffic is much more than in Minimal. The schedule, as is, is not adaptive to traffic demand and there are instances where nodes suffer from a high number of queued packets [5]. Work has been done in order to provide Orchestra with additional features, *e-TSCH-Orchestra* [6], specifically in adapting its schedule based on traffic demand, reducing end-to-end delay and differentiates traffic. This, however, requires more slots in its schedule thus increasing each node's duty cycle.

Average latency appears to be significantly better in TSCH rather than GinMAC. This could be explained by GinMAC's TDMA frame size (760ms) compared to TSCH's frame size(10ms and 160ms). A node in GinMAC waits a lot longer than TSCH in order to send any data that are available to send.

Our scenarios consisted of a small number of nodes, 16 in particular, in order to allow GinMAC to work with its maximum load with a 3-2-1 tree configuration. Minimal shows that its the preferred choice for a network of this size. Orchestra shows that for larger network sizes it is a better choice in terms of PDR and PRR. Also, Orchestra shows that is has the lowest duty cycle out of all the combinations tested [4].

In the end, the prefered protocol choice when designing a WSN is ultimately determined by the nature of the network and its requirements. GinMAC provides guaranteed data delivery within a certain time frame. TSCH , with Minimal or Orchestra, provides a less complex implementation of a WSN , also proven by its duty cycle. This is more suited towards simpler network designs with light time restraints.

6.2 Future work

As we have mentioned in our results, GinMAC provided us with the best overall performance in any of our tests, excluding CSMA. However, further work needs to be established in DTC's implementation. Referring to our *Node Deadlock* case, DTC should be able to determine the best suited node to parent relationship for each node in order to provide service to as many nodes possible. This also means the ability to change nodes between parents , and their corresponding slots, in order to make way for new slots that need to access the network.

In order to come to a full conclusion regarding TSCH and GinMAC, they must also be tested under known traffic patterns. So far, GinMAC provided us with the best overall method for randomized traffic patterns. The next step is to test them with predefined schedules, without the use of DTC or any TSCH related scheduling algorithm. TSCH dedicated should provide us with interesting results as TSCH has less radio duty cycle than GinMAC because of it's slot handling. Also, end-to-end delay could be lower in TSCH when exploiting it's channel hopping technique whereas GinMAC only allows communication between two nodes in one slot.

Bibliography

[1] Documentation: orchestra. http://https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-Orchestra. Accessed: 2019-05-02.

[2] Documentation: rpl. http://https://github.com/contiki-ng/contiki-ng/wiki/Documentation:-RPL. Accessed: 2019-04-20.

[3] Platform cooja. http://https://github.com/contiki-ng/contiki-ng/wiki/Platform-cooja. Accessed: 2019-05-02.

[4] M. Mohamadi, B. Djamaa, and M. R. Senouci. Performance evaluation of tsch-minimal and orchestra scheduling in ieee 802.15.4e networks. In *2018 International Symposium on Programming and Systems (ISPS)*, pages 1–6, Apr. 2018. DOI: 10.1109/ISPS.2018.8379007.

[5] S. Rekik, N. Baccour, M. Jmaiel, and K. Drira. A performance analysis of orchestra scheduling for time-slotted channel hopping networks. *Internet Technology Letters*, 1(3):e4, 2018. DOI: 10.1002/itl2.4. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/itl2.4. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/itl2.4.

[6] S. Rekik, N. Baccour, M. Jmaiel, K. Drira, and L. A. Grieco. Autonomous and traffic-aware scheduling for tsch networks. *Computer Networks*, 135:201–212, 2018. ISSN: 1389-1286. DOI: https://doi.org/10.1016/j.comnet.2018.02.023. URL: http://www.sciencedirect.com/science/article/pii/S1389128618300963.

[7] S. Duquennoy, A. Elsts, B. A. Nahas, and G. Oikonomo. Tsch and 6tisch for contiki: challenges, design and evaluation. In *2017 13th International Conference on Distributed Computing in Sensor Systems (DCOSS)*, pages 11–18, June 2017. DOI: 10.1109/DCOSS.2017.29.

[8] V. X. Ed., P. K., and T. Watteyne. Minimal IPv6 over the TSCH Mode of IEEE 802.15.4e (6TiSCH) Configuration. RFC 8180, RFC Editor, May 2017. DOI: 10.17487/RFC8180. URL: https://www.rfc-editor.org/info/rfc8180.

[9] J. F. Kurose and K. W. Ross. *Computer Networking*. Pearson, 7th edition, 2017.

[10] S. Duquennoy, B. Al Nahas, O. Landsiedel, and T. Watteyne. Orchestra: robust mesh networks through autonomously scheduled tsch. In *Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems*, SenSys '15, pages 337–350, Seoul, South Korea. ACM, 2015. ISBN: 978-1-4503-3631-4. DOI: 10.1145/2809695.2809714. URL: http://doi.acm.org/10.1145/2809695.2809714.

[11] Standard for low-rate wireless networks. *IEEE Std. 802.15.4(2015)*, 2015.

[12] T. Watteyne, M.Palattella, and L. Grieco. Using IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) in the Internet of Things (IoT): Problem Statement. RFC 7554, RFC Editor, May 2015. DOI: 10.17487/RFC7554. URL: https://www.rfc-editor.org/info/rfc7554.

[13] Z. Zinonos. *Wireless Sensor Networks Mobility Management: A Performance Control Approach*. PhD thesis, University of Cyprus, Sept. 2013.

[14] W. T. Ed., T. P. Ed., B. A. H. J. K. R., L. P., P. K., S. R., V. JP., and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550, RFC Editor, Mar. 2012. DOI: 10.17487/RFC6550. URL: https://www.rfc-editor.org/info/rfc6550.

[15] Z. Zinonos, V. Vassiliou, C. Ioannou, and M. Koutroullos. Dynamic topology control for wsns in critical environments. In *2011 4th IFIP International Conference on New Technologies, Mobility and Security*, pages 1–5, Feb. 2011. DOI: 10.1109/NTMS.2011.5720652.

[16] T. O'Donovan, J. Brown, U. Roedig, C. J. Sreenan, J. do O, A. Dunkels, A. Klein, J. Sa Silva, V. Vassiliou, and L. Wolf. Ginseng: performance control in wireless sensor networks. In *2010 7th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–3, June 2010. DOI: 10.1109/SECON.2010.5508206.

[17] S. P., B. J., and R. U. Time-critical data delivery in wireless sensor networks. *Rajaraman R., Moscibroda T., Dunkels A., Scaglione A. (eds) Distributed Computing in Sensor Systems. DCOSS 2010.*, 6131, 2010.

[18] N. Tsiftes, J. Eriksson, and A. Dunkels. Poster abstract: low-power wireless ipv6 routing with contikirpl. In *:* 9th edition, 2010.

[19] I. Akyildiz, W.Su, Y.Sankarasubramaniam, and E.Cayirci. Wireless sensor networks: a survey. *Computer Networks*, 38(4):393–422, 2002.