



VRIJE  
UNIVERSITEIT  
BRUSSEL



# PROJECT 2

Unslotted CSMA-CA Vs TSCH Orchestra

Maarten Dequanter / Imad Mesbih

May 20, 2025

# Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Hypothesis and reasoning</b>	<b>2</b>
<b>3</b>	<b>Experimental setup</b>	<b>2</b>
3.1	Simulation setup requirements . . . . .	2
3.2	Fully Automated Testing procedure . . . . .	4
<b>4</b>	<b>Results and Discoveries</b>	<b>6</b>
4.1	PDR (Packet Delivery Ratio) Breaking Point . . . . .	6
4.1.1	CSMA performance analysis . . . . .	7
4.1.2	TSCH Orchestra – High Variance and the Need for Robust Metrics . . . . .	8
4.2	Impact on latency . . . . .	9
4.2.1	CSMA . . . . .	9
4.2.2	TSCH orchestra . . . . .	9
4.3	Throughput analyses (bits/second) . . . . .	10
4.3.1	CSMA . . . . .	10
4.3.2	TSCH . . . . .	10
4.4	Energy Benefit of TSCH over CSMA . . . . .	10
4.5	Conclusion . . . . .	13
<b>5</b>	<b>Behavior under Disturbances</b>	<b>13</b>
5.1	Protocol Behavior . . . . .	13
5.1.1	CSMA under Disturbances . . . . .	13
5.1.2	TSCH with Orchestra under Disturbances . . . . .	13
5.2	Expected Results . . . . .	14
5.3	Experimental Setup . . . . .	14
5.4	Observations and Interpretation . . . . .	15
5.5	Extended Disturbance Experiment: Moving Multiple Senders . . . . .	15
5.5.1	Observations and Interpretation . . . . .	16
<b>6</b>	<b>Exponential Backoff</b>	<b>16</b>
6.1	Exponential Backoff in CSMA-CA . . . . .	17
6.2	Backoff Mechanisms in TSCH . . . . .	18
6.3	Impact on Performance . . . . .	18
<b>7</b>	<b>Impact of Slotframe Size on Performance</b>	<b>19</b>
<b>8</b>	<b>Bandwidth, Packet Size, and Channel Usage in TSCH vs CSMA</b>	<b>20</b>
8.0.1	Example Per-Node Throughput Calculation for CSMA . . . . .	20
8.0.2	Packet Delivery Ratio (PDR) Measurement in TSCH . . . . .	20
<b>9</b>	<b>Future academic work</b>	<b>21</b>
9.1	TSCH Synchronization lost at Low Traffic . . . . .	21
9.2	Latency Variance and PDR Recovery . . . . .	22
<b>10</b>	<b>APPENDIX</b>	<b>23</b>

# 1 Introduction

In this project, we aim to systematically compare two Medium Access Control (MAC) protocols used in Wireless Sensor Networks (WSNs): **Unslotted CSMA-CA** and **TSCH with Orchestra scheduling**. Our primary goal is to evaluate and contrast their performance in terms of **Packet Delivery Ratio (PDR)**, **latency**, and **throughput** under varying network loads and conditions.

The outcomes of this project not only highlight the theoretical strengths and weaknesses of each MAC protocol but also reveal practical implementation challenges that arise in real-world network deployments.

## 2 Hypothesis and reasoning

We chose to simulate a wireless sensor network (WSN) where 20 sender nodes transmit data to a single **centralized sink node**, namely node 16. (Figure 1). This setup allows us to measure the throughput and its breaking point, PDR and latency of messages received from all sender nodes. As mentioned in paper (Shah & Akan, 2014) we took 20 sender nodes, to see the impact on both **CSMA Scheduler** and **TSCH with Orchestra scheduler**.

The network forms a **hybrid topology** that visually resembles a **multi-tier star**, but functionally behaves like a mesh network, where nodes that are not in direct range of the sink must rely on intermediate forwarding nodes to deliver their data. In this simulation, nodes such as node 3, 6, 25, and 26, which are located closer to the sink (within the inner green region), play a crucial role as **relay nodes**, forwarding messages from more distant nodes toward the sink.

This setup, with many nodes operating within each other's interference range, significantly increases the likelihood of packet collisions and queue saturation. This environment is ideal for examining the behavior and impact of the **exponential backoff** mechanism and the impact of disturbance nodes under high contention.

## 3 Experimental setup

### 3.1 Simulation setup requirements

We need to measure the **packet delivery ratio (PDR)**, **throughput**, and **latency** introduced by both **Unslotted CSMA** and **TSCH with the Orchestra scheduler**. This is achieved by systematically increasing the sending rate (in messages per minute). To ensure accurate and meaningful results, the following requirements were implemented:

- **Precise timing measurement from sender to receiver:** To achieve this, each message includes a unique string containing the sender's IPv6 address and a sequence number. For example: 'Msg fd00::202:2:2:2 6'. The simulation logs the exact time when the message is sent, allowing precise end-to-end latency calculation.
- **netto vs. bruto messages** Because there might be situations where a node is unable to send, for instance (Tx queue saturation), This has not only impact on the way how we calculate the PDR, but also the latency. By monitoring the effective sent (by testing queue message) messages we can do better analysis.
- **Sufficient number of messages:** Each sender node transmits 100 messages per simulation bin to generate a statistically significant number of data points.

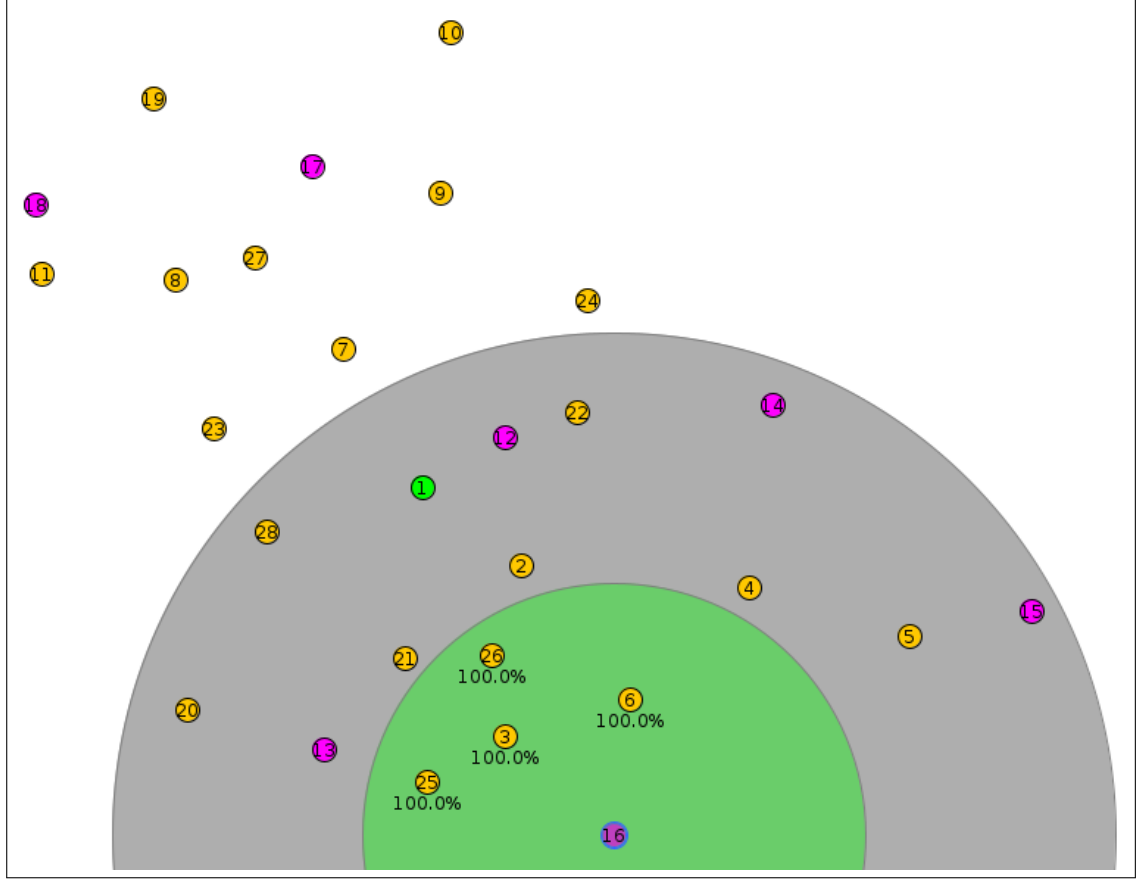


Figure 1: Final setup: 20 nodes sending to node 16

- Even distribution of message sending:** To avoid artificial peaks caused by unwanted synchronized transmissions between our 20 sending nodes, we introduced a `JITTER_PERCENT` of 100% in the `sender-node.c` implementation. This ensures that messages are sent in a randomized manner within the defined sending interval, providing a more realistic traffic pattern. Without this implementation we have peaks of 20 times the sending sequence, because all 20 sending nodes can potentially send at almost exact time.
- Measurements during steady-state conditions:** To improve consistency, data (Messages) sending only begins once the network has fully converged. A helper script (`checkNetworkIsBuilt.py`) was used to detect when all nodes had obtained an RPL rank, marking as the starting point for measurements.
- Handling randomness with Orchestra:** Prior research indicates that TSCH combined with the Orchestra scheduler can lead to complex and sometimes unpredictable behavior in dense IoT networks. Therefore, we chose to capture detailed logs from the moment the network is built until the last message is received, because via tests we noticed that the joint order of the nodes impacts the whole simulation performance. This comprehensive data collection approach ensures that any anomalies or transient behaviors can be fully analyzed per node if needed. As an example you can see the collected data for a TSCH

bin, created by script TSCH\_logfileAnalyser.py in Table 1.

Table 1: Per-node performance under TSCH with reduced throughput and queue metrics

Sender Node	Latency (ms)	Sent	Confirmed	Received	Throughput (%)	Sendrate (Bps)	Avg Hops	Lines	Queue Full
10	805.05	100.00	100.00	39.00	39.00	1.62	5.00	4881.00	0.00
11	803.84	100.00	100.00	48.00	48.00	2.07	4.00	7095.00	0.00
19	807.29	100.00	100.00	40.00	40.00	1.91	5.00	6925.00	0.00
2	356.79	100.00	95.00	77.00	81.10	1.55	2.00	17 601.00	<b>235</b>
20	478.36	100.00	74.00	33.00	44.60	1.51	4.00	13 703.00	<b>656</b>
21	493.10	100.00	100.00	44.00	44.00	2.11	3.00	11 030.00	0.00
22	471.90	100.00	100.00	45.00	45.00	2.15	3.00	8832.00	0.00
23	633.15	100.00	100.00	38.00	38.00	0.64	3.16	14 084.00	0.00
24	656.22	100.00	100.00	39.00	39.00	1.83	3.00	9817.00	0.00
25	481.61	100.00	100.00	37.00	37.00	0.64	3.51	11 015.00	<b>147</b>
26	321.68	100.00	46.00	40.00	87.00	1.84	1.00	36 373.00	<b>1605</b>
27	700.33	100.00	100.00	38.00	38.00	1.81	4.00	8501.00	0.00
28	740.57	100.00	100.00	47.00	47.00	2.20	3.00	8899.00	0.00
3	444.49	100.00	100.00	76.00	76.00	1.57	3.00	12 782.00	<b>298</b>
4	779.95	100.00	100.00	40.00	40.00	1.63	3.30	8579.00	0.00
5	742.05	100.00	100.00	47.00	47.00	1.96	4.32	5816.00	0.00
6	255.29	100.00	73.00	72.00	98.60	1.51	1.00	12 693.00	<b>199</b>
7	591.51	100.00	100.00	37.00	37.00	1.55	3.00	9880.00	0.00
8	776.51	100.00	100.00	41.00	41.00	0.63	4.24	8708.00	0.00
9	593.02	100.00	100.00	44.00	44.00	1.75	4.00	7985.00	0.00
<b>Mean</b>	596.63	100.00	94.00	46.00	48.80	1.62	3.33		

### 3.2 Fully Automated Testing procedure

To ensure realistic results with 95% confidence intervals, we require at least 30 simulation bins for each sending rate. Given the requirement for end-to-end analytics, this results in a minimum of 300 simulation bins and about 15GB data collection.

To facilitate this, we implemented two scripts: `run-coojaTSCH.py` and `run-coojaCSMA.py`, which automatically launch the Cooja environment with or without a GUI. These scripts configure the correct sending rate(`sender-node.c`), simulation time (`coojalogger.js`), project settings (`project-conf.h`), and debug logging parameters. The input parameters include a list of sending rates and the number of bins. This setup allows us to execute multiple simulation bins with different sending rates at various stages of the analysis. The different sending rates for both CSMA and TSCH Orchestra Scheduler can be found in table 2. As you will notice other sending rates are used because of the different behaviour in our complex multi-hop setup.

Table 2: Send rates used in the experiments for CSMA and TSCH (messages/min)

Scheduler	Send Rates (messages/min)
Unslotted CSMA-CA	40, 43, 46, 50, 55, 60, 67, 75, 86, 100, 120, 150, 200, 300, 600
TSCH with Orchestra	3, 4, 6, 7, 8, 9, 10, 12, 15, 20, 30, 60

For TSCH we used adapted settings for the `QUEUEBUF_CONF_NUM = 64`, so in our testing scenario we simulate devices with 256KB RAM memory like the nRF52840.

The `TSCH_QUEUE_CONF_MAX_PACKETS_PER_NEIGHBOR` was set to 16(standard 8). This was confirmed by in file `'contiki-ng/os/net/mac/tsch/tsch.c'` from line 1153 to 1159.

With those changes we could confirm better performance as the queues can be managed better in the heavily loaded WSN. No other settings were made in the `project-conf.h` file.

To ensure each test is as realistic as possible, we use a **generated random seed**, which is configured in the COOJA environment's CSC file. Without this, we would always obtain

identical results, since the random number generators would otherwise use fixed values. Both schedulers TSCH Orchestra and CSMA share the same COOJA configuration file.

The script automatically loops over all requested sending rates and bins. After each test bin for a given sending rate, a CSV record line is generated in the format: (File, Timing, Batch, Sender, End-to-End latency (ms), Sent, Confirmed, Received, Throughput, Sendrate (Bps), Avg Hops). The corresponding log file (e.g., CSMA\_2\_30.testlog) is saved in a dedicated logging folder for further analysis.

This approach enabled us to run extensive tests over longer periods without delay or manual intervention. To monitor the simulation process, a live dashboard was developed, which is particularly useful for interrupting or adjusting the process as needed. This is shown in Figure 2.

#### Live TSCH Queue Fill, Reception & Latency (code/analyses/COOJA.testlog)

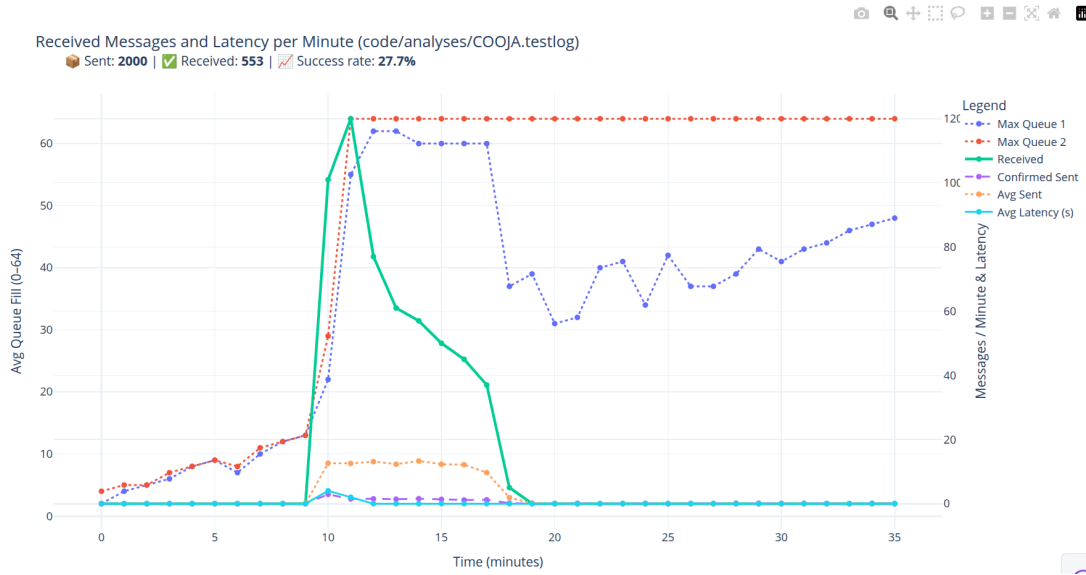


Figure 2: Live dashboard to monitor simulation process

Once all desired simulations have been executed and the corresponding log files and result summaries have been saved, the analysis phase can begin using various custom Plotly-based scripts. These scripts take full advantage of Plotly's interactive features, enabling users to zoom, pan, highlight specific time intervals, and toggle parameters on and off with ease. This interactivity not only makes complex datasets more accessible but also facilitates faster pattern recognition, anomaly detection, and visual comparisons across configurations.

This is a limited list of scripts for both TSCH and CSMA that have been created, more example picture and link to GitHub pages are shown in the appendix:

- `dashqueue.py`:  
This analyzer script, visualizes key metrics such as queue fill, message reception, end-to-end latency, and Trickle resets over time.
- `dashqueueindividual.py`:  
It visualizes queue fills of the specific relay nodes (3,25,26 and 6) next to the receivers node. Other nodes can easily be changed in the code if needed.

- `trickleTimerPerMinute.py`: This script displays the Trickle timer counts per minute. When the network is stable, the timer backs off and messages are sent less frequently to conserve energy and bandwidth, we used this script to see if there were issues.
- `TSCHConfidenceIntervals.py` and `CSMAConfidenceIntervals.py`: This chart shows the impact of increasing TSCH send rates (in messages per minute) on end-to-end latency, packet delivery ratio (PDR), and throughput, with 95% confidence intervals via bar charts. This gives a good overview per bin. Different versions of these files are available on GitHub
- `TSCHConfidenceIntervalsContinuous.py` and `CSMAConfidenceIntervalsContinuous.py`: This chart also shows these 95% confidence intervals, but with a linear scale of messages send per minute. This way it is easy to analysis en detect breaking points

## 4 Results and Discoveries

For both CSMA and TSCH with the Orchestra scheduler, we conducted at least 30 simulation runs ("bins") for each configured send rate (see Table 2). In total, this resulted in approximately 694 distinct testlog files, amounting to 15.8 GB of data. The complete generation of these simulation bins required over 10 hours of processing time. Despite the computational cost, this extensive dataset provides a solid foundation for thorough and reliable analysis.

### 4.1 PDR (Packet Delivery Ratio) Breaking Point

The breaking point of the Packet Delivery Ratio (PDR) refers to the send rate at which network reliability begins to consistently degrade—typically due to congestion, interference, or scheduling constraints. In wireless sensor network (WSN) research, a PDR of at least **95%** is often considered acceptable. Values below **90%** are indicative of performance degradation, while a PDR below **80%** generally marks a critical threshold for reliable communication Lauwens et al., 2010.

Given that our setup involves a heavily loaded WSN, we define the breaking point at a PDR of **80%**.

In the case of CSMA, we observed excellent performance across a wide range of send rates (see Figure 3). The PDR only starts to decline after an average per node rate of approximately **55 messages/minute**, reaching the **80%** threshold around **65 messages/minute** per senders node. Even at extremely high traffic rates such as **140 messages/minute**, the PDR still stabilizes around **20%**.

By contrast, when using the TSCH Orchestra scheduler (Figure 4), the situation is significantly different. The maximum observed mean PDR reaches only **81.3%**, and it drops below the **80%** threshold at an avg. senders send rate as low as **7 messages/minute**. This is approximately 'ten times lower' than the breaking point observed with unslotted CSMA.

These results highlight that, in high-density environments, CSMA outperforms TSCH Orchestra in terms of maintaining a high PDR under load.

However, it's also important to note that TSCH Orchestra exhibits much higher variability than CSMA. This is due to fluctuations in node behavior and network initialisation, scheduling alignment, and queue overflows. As such, using only the mean PDR is insufficient to fully capture protocol performance. These issues—and the motivation for incorporating median and min/max statistics—are discussed in the following section.

### PDR with 95% Confidence Interval (CSMA)

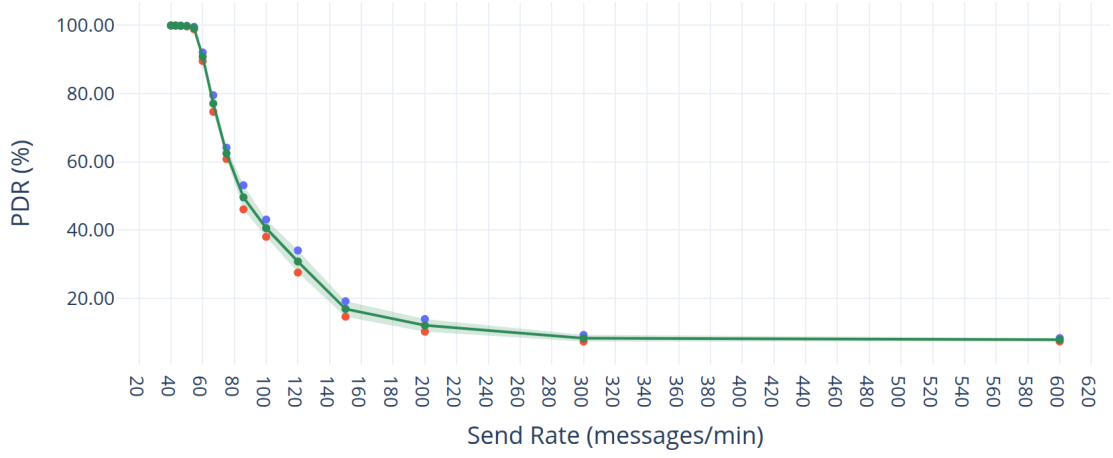


Figure 3: CSMA PDR (package delivery ratio)

### TSCH PDR (%) by Send Rate

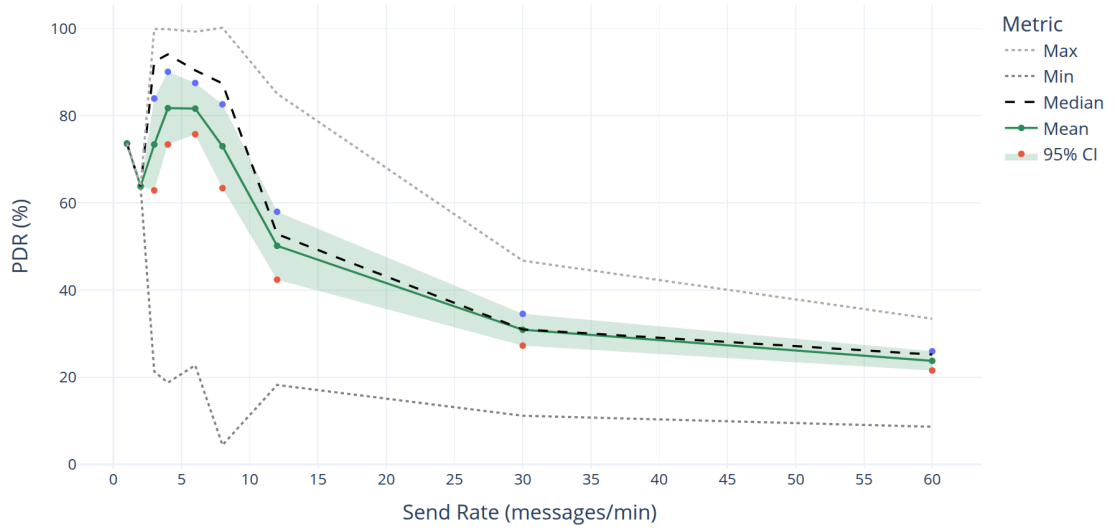


Figure 4: TSCH with Orchestra PDR (package delivery ratio)

#### 4.1.1 CSMA performance analysis

Figure 5 illustrates the performance of CSMA under near-saturation conditions. With a success rate of 91.3%, the protocol maintains high reliability even as the queue length remains close to the configured upper limit of 64. As you can see on this Figure, when the queue is below 60, at around second 650, it is clear that received messages follows the sent messages exactly. This indicates that the system is operating at full capacity. The slight and temporary drops in the



number of received messages suggest moments of congestion or collision, but the protocol quickly recovers without significant long-term degradation. These results confirm that CSMA handles high traffic volumes effectively up to this threshold, but further increasing the load may cause queue overflows and higher packet loss.

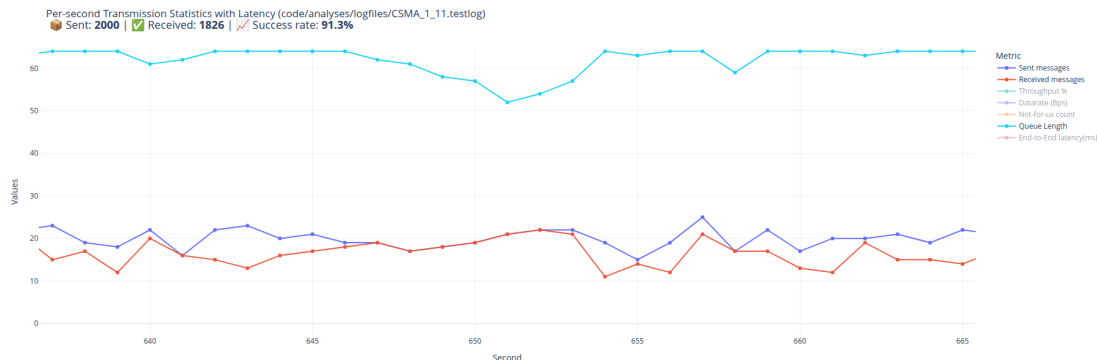


Figure 5: CSMA: performing at near-saturation conditions

#### 4.1.2 TSCH Orchestra – High Variance and the Need for Robust Metrics

While CSMA was very stable in the examined Cooja environment, TSCH Orchestra exhibited significantly more variability and unpredictability. This instability motivated the inclusion of additional statistical representations—namely the median and min/max values—alongside the mean in our analysis.

The necessity of this expanded view becomes evident in Figure 4, which shows that the maximum **median** PDR reaches **94%** at a send rate per node of **4 messages per minute**, whereas the **mean** PDR for the same rate is only **81.74%**. This large gap between mean and median highlights that while TSCH Orchestra can perform well in many instances, the average is skewed downward by several outlier runs in which critical relay nodes failed.

Such failures typically occur when the forwarding queues of these relay nodes overflow. As described earlier, we use a `TSCH_QUEUE_CONF_MAX_PACKETS_PER_NEIGHBOR` of 16. Figure 1 illustrates the key relay nodes **25, 3, 26, and 6** which forward traffic to the receiver node 16 and form **bottlenecks** in the topology.

By examining the test logs near the critical PDR drop-off around the 80% threshold, we clearly observe how queue saturation in these nodes leads to increased message loss (see Figure 6, generated with `dashqueuesIndividual.py`). For instance, around minute 25, Q25 exceed the configured maximum of 16 packets, triggering immediate losses. Earlier, around minute 12, node Q26 queue temporarily spikes, but forwarding still succeeds thanks to the available capacity in other relay nodes.

These observations confirm that relying solely on mean PDR obscures the protocol’s peak performance and masks critical reliability issues that only appear under specific network conditions. Including median and min/max values is therefore essential to fully capture TSCH Orchestra’s behavior.

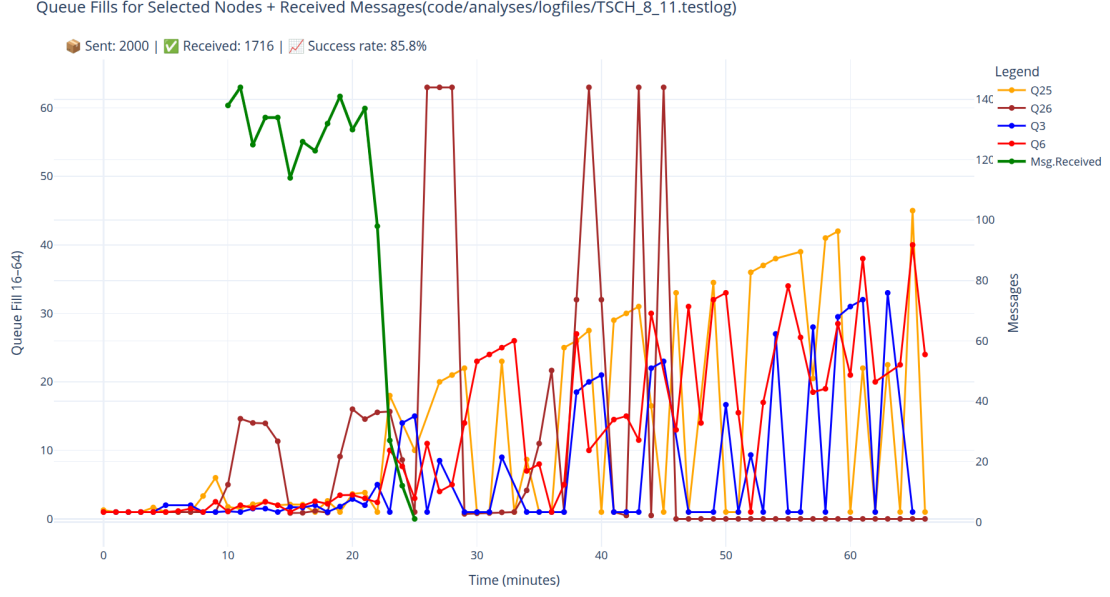


Figure 6: Relay nodes 25,3,26 and 6 queues overflow in TSCH

## 4.2 Impact on latency

To understand the results we need to clearly explain how the End-to-end latency is measured in our tests. When a message sending procedure is initiated, a message is printed in the cooja.testlog file: 600333000 8 Sending message: 'Msg fd00::208:8:8:2' to fd00::210:10:10:10 It includes a unique string containing the sender's IPv6 address and a sequence number. Then the time is measured when exactly this message arrives at the receivers node 16, by evaluating the message content.

Some time later the receivers node 16 receives that message: 600542336 16 Data received from fd00::208:8:8:8 on port 1234 from port 1234 in 4 hops with datalength 22: 'Msg fd00::208:8:8:2' The simulation logs the exact time when the message is sent, allowing precise end-to-end latency calculation. In this case it is 209.336 ms.

### 4.2.1 CSMA

A minimal latency of 228,57ms is observed at a send rate of 40 messages/minute per node. As observed in the PDR graph of CSMA (Figure 3), a noticeable change also occurs in the latency vs. message rate graph (Figure 7) around a send rate of 55 messages per minute per node. At 60 msg/minute per node it reaches 3,75 s. From approximately 140 messages per minute onward, the latency begins to increase linearly. Once again, we can conclude that this behavior strongly correlates with the trend observed in the PDR. The changes and high variances at 8, 12 and 30 msg/min are also seen in the PDR graph.

### 4.2.2 TSCH orchestra

A minimal latency of 540,13 ms is observed at 3 msg/minutes, but increases almost linear up to 18 seconds at 60 msg/second. As we can see in Figure 8, we have a high variance around at

sending rates of 8, 12 and 30 messages/minute. This also occurs at the moments when the PDR drops. It drops, in other words, the latency is highly correlated with the PDR.

### 4.3 Throughput analyses (bits/second)

In this performance analysis, the **throughput** is computed per sender node based on the amount of data that is successfully received by the sink node. For each message transmission, the script records the message content and timestamp. Upon reception, the message is matched with its corresponding transmission event, and the delay, hop count, and payload size are recorded.

The throughput is calculated using the following formula:

$$\text{Throughput (bits/s)} = \frac{\text{Total received bytes} \times 8}{\text{Time span (s)}}$$

where the time span is defined as the interval between the first recorded transmission attempt and the last successful reception of a message for each individual sender node.

This method results in a **net throughput** metric, as it only considers data that was effectively delivered. It excludes retransmissions, lost packets, or unconfirmed sends, and therefore accurately reflects the usable data delivery capacity of the network under varying load conditions.

#### 4.3.1 CSMA

We can see on Figure 9, starting a sending rate of 40 messages/minute, the throughput start increasing up to 55 messages/minute. We see the same kind of figure in the latency graph. We can explain it as follows: As you increase the sending rate, more data is successfully delivered per second. This leads to a rising throughput (bits/s), up to the network's capacity.

Then latency Increases, the higher send rates fill up queues at each node. Packets wait longer in the buffer before being transmitted. This results in increased end-to-end latency, even though packets are still getting through.

PDR (Packet Delivery Ratio) Starts Dropping. Once the network approaches its saturation point, queues begin to overflow. When buffers are full, new incoming packets are dropped. This means fewer messages are successfully received, and the PDR declines.

#### 4.3.2 TSCH

As you can see when we compare the throughput based on the sending rate between CSMA on Figure 9 and for TSCH with Orchestra on Figure 10, we have a completely different behavior. The difference in the throughput evolution graph for TSCH compared to CSMA is due to how TSCH with Orchestra schedules transmissions versus how CSMA handles access to the channel. TSCH uses a **deterministic schedule based on slotframes** and **avoids collisions** via time-slot assignments. Even at high send rates, if slots are available and queues don't overflow, throughput continues to increase because nodes transmit in isolated time slots.

The TSCH graph shows no throughput saturation yet, because the scheduling and queuing system still handles the load efficiently up to 60 msgs/min. On the other hand the PDR starts dropping at 5-6 msgs/min. In TSCH the PDR is a more realistic indicator on evaluating if a WSN works well.

### 4.4 Energy Benefit of TSCH over CSMA

It is not asked in the task description, but as observed during the experiment, the TSCH protocol with the Orchestra scheduler demonstrates a significant energy efficiency advantage over CSMA.

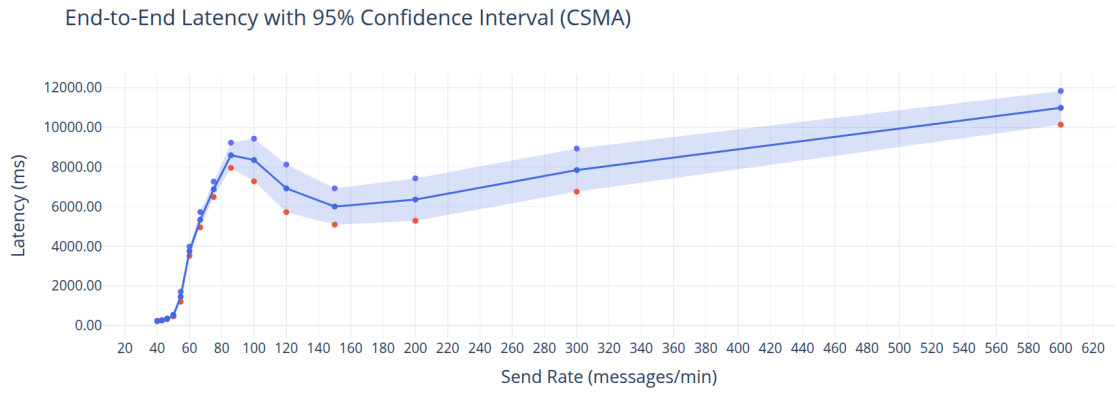


Figure 7: End-to-End Latency with 95 Confidence Interval (CSMA)

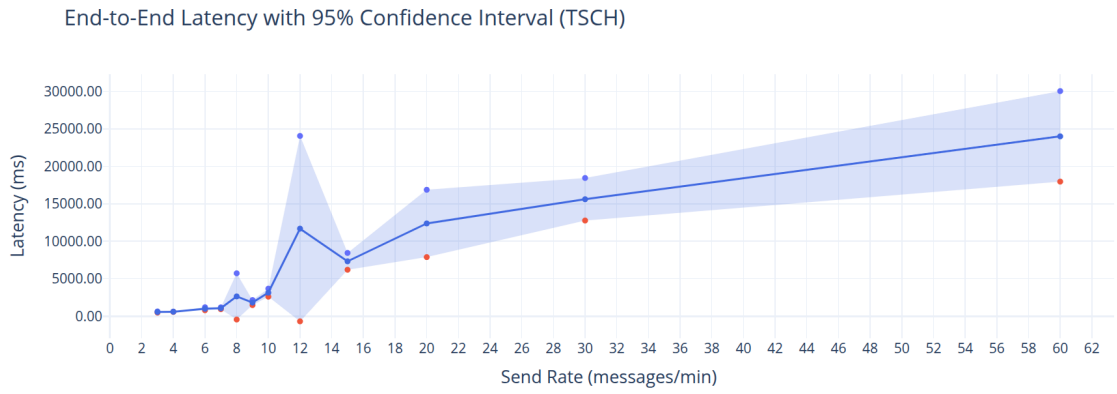


Figure 8: End-to-End Latency with 95 Confidence Interval (TSCH)

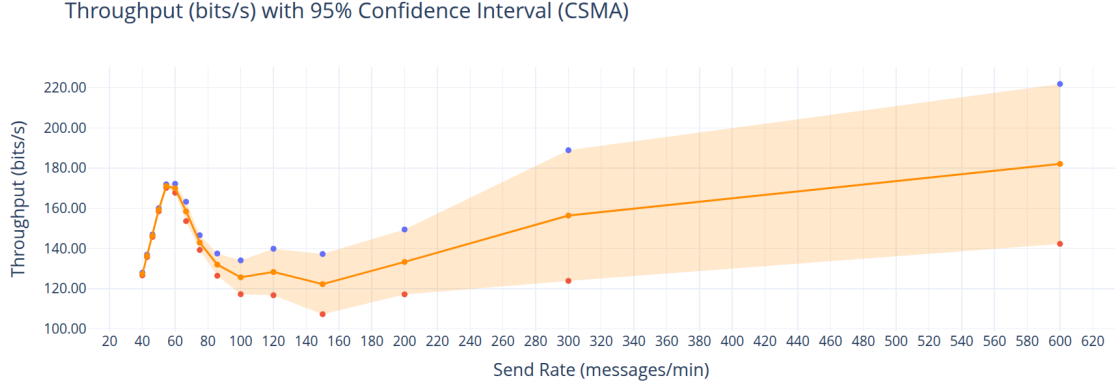


Figure 9: Throughput CSMA with 95 confidence intervals

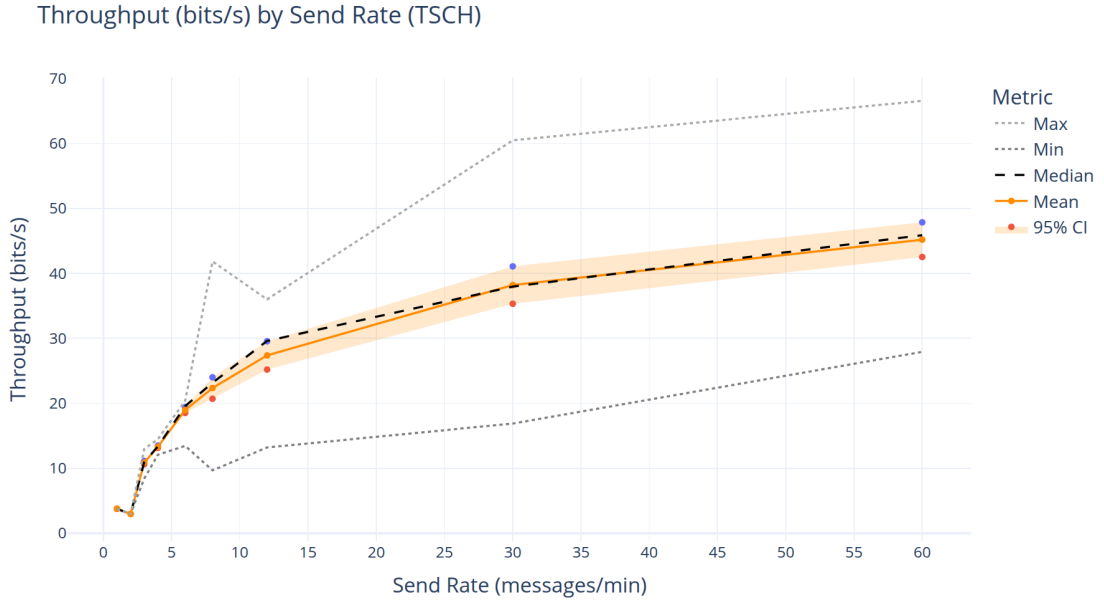


Figure 10: Throughput TSCH with 95 confidence intervals

The average radio-on time for TSCH was measured at only 3.59% (see Figure 18 in the appendix), compared to a constant 100% radio-on time for CSMA (see Figure 19 in the appendix) throughout the entire process.

It is noteworthy that nodes located at the edges of the simulation area exhibit higher radio-on durations, particularly at the beginning of the simulation. This behavior is likely due to increased difficulty in joining the network, which may also influence the overall success rate. This phenomenon needs further investigation in future work.

## 4.5 Conclusion

This section compared Unslotted CSMA-CA and TSCH with Orchestra under heavy network load in a wireless sensor network. CSMA outperformed TSCH in terms of PDR and throughput, sustaining reliable communication up to 65 messages/second. TSCH, in contrast, dropped below the 80% PDR threshold at only 7 messages/minute per senders node. These are values per senders node. Because we have 20 senders nodes, CSMA can handle 1300 messages/minute and TSCH 140 messages per minute.

We revealed high variance in TSCH performance. While its **mean** PDR was modest, the **median** peaked at 94%, showing that averages alone don't reflect true potential. Adding median and min/max plots provided critical insights into protocol behavior under stress.

Latency and throughput analyses confirmed CSMA's robustness and TSCH's sensitivity to queue overflow and relay node congestion. Nonetheless, TSCH offered a clear energy advantage, with radio-on time below 4%.

In the chosen scenario CSMA is better suited for high-throughput scenarios, while TSCH requires careful tuning and deeper analysis but offers strong energy efficiency and scheduling determinism.

## 5 Behavior under Disturbances

In wireless sensor networks, **disturbances** refer to any unexpected condition that disrupts the normal flow of communication. These disturbances can include interference, node mobility, dynamic topology changes, external noise, or congestion due to overloaded nodes. Such disruptions can have significant impacts on network performance metrics such as latency, packet delivery ratio, and energy efficiency.

### 5.1 Protocol Behavior

#### 5.1.1 CSMA under Disturbances

**Carrier Sense Multiple Access (CSMA)** handles disturbances reactively. It uses an *exponential backoff mechanism*—when the medium is busy or a collision is detected, the node waits for a random backoff period before retrying. This approach allows the protocol to quickly adapt to interference or dynamic changes in the environment. However, in dense networks or under heavy load, CSMA tends to suffer from increased contention and collisions, which can degrade throughput and increase latency.

#### 5.1.2 TSCH with Orchestra under Disturbances

**Time Slotted Channel Hopping (TSCH)** in combination with **Orchestra scheduling** offers a proactive and deterministic approach. Communications occur in predetermined time slots across multiple frequencies, significantly reducing collisions and improving robustness against interference. While this structured behavior enhances reliability, it also introduces rigidity. In the face of disturbances such as topology changes or overloaded relay nodes, Orchestra's fixed schedule may not adapt quickly enough, potentially causing transmission queue overflows and packet losses.

## 5.2 Expected Results

Based on the known characteristics of CSMA and TSCH with Orchestra, we hypothesize the following outcomes when subjecting the network to a controlled disturbance:

### CSMA:

- CSMA is expected to react quickly to disturbances due to its reactive backoff mechanism.
- However, in scenarios with increased traffic or sudden topology changes, CSMA is likely to experience increased collisions and retransmissions, especially in denser areas of the network.
- Therefore, we hypothesize that CSMA will show degraded performance in terms of packet delivery ratio and latency following the disturbance, particularly at higher message rates.

### TSCH with Orchestra:

- Due to its scheduled and synchronized nature, TSCH with Orchestra is expected to be robust against transient interference and avoid most collisions.
- However, because Orchestra’s schedule is static and does not adapt dynamically to changes in traffic or topology, we anticipate that queue build-ups or overflows may occur at relay nodes following a disturbance.
- Thus, we hypothesize that TSCH with Orchestra may experience a temporary drop in packet delivery or an increase in latency, especially if the moved node is part of a critical routing path.

### Overall Expectation:

- At low message rates, both CSMA and TSCH with Orchestra are expected to tolerate the disturbance with minimal performance degradation.
- At higher message rates, CSMA may suffer from congestion and collisions, while TSCH may face limitations from fixed scheduling and queue constraints.

This hypothesis provides a baseline for interpreting the outcomes of our simulation experiments.

## 5.3 Experimental Setup

To explore how CSMA and TSCH with Orchestra react to disturbances, we conducted simulations in Contiki-NG under semi-controlled conditions. While Contiki-NG includes support for “disturber nodes,” we found that this functionality did not behave as expected. As an alternative, we introduced a topology disturbance by **moving a node (Node 25)**, which was initially positioned close to the sink, **further away after 300 seconds** of simulation time. This disturbance occurred after the network had stabilized and routing paths were formed.

We ran simulations in **three batches** with different message rates to examine the sensitivity of each protocol to traffic load. The message rates were chosen to be **close to the breaking point** that we had identified in earlier experiments.

Although the number of batches is limited, this is sufficient to observe trends in protocol behavior.

## 5.4 Observations and Interpretation

Unexpectedly, the introduced disturbance did not lead to a significant drop in performance for either protocol, as can be seen in Figures 11 and 12. In some cases, the performance remained roughly the same, and in others, it was even slightly better after the disturbance. Several possible explanations can account for this outcome:

- **Routing Stabilization:** The network had sufficient time to establish and optimize routing paths before the disturbance, allowing the RPL protocol to recover smoothly from the topology change.
- **Limited Impact Scope:** Moving only a single node (Node 25), especially if it was not a critical relay for many routes, may not have significantly altered the network’s overall structure or performance.
- **Local Traffic Relief:** If multiple (in our case four) senders were positioned close to the receiver, removing one of them could have reduced contention and localized congestion near the sink. This might have unintentionally improved channel access and queue behavior in that area, offsetting the expected negative effects of the disturbance.

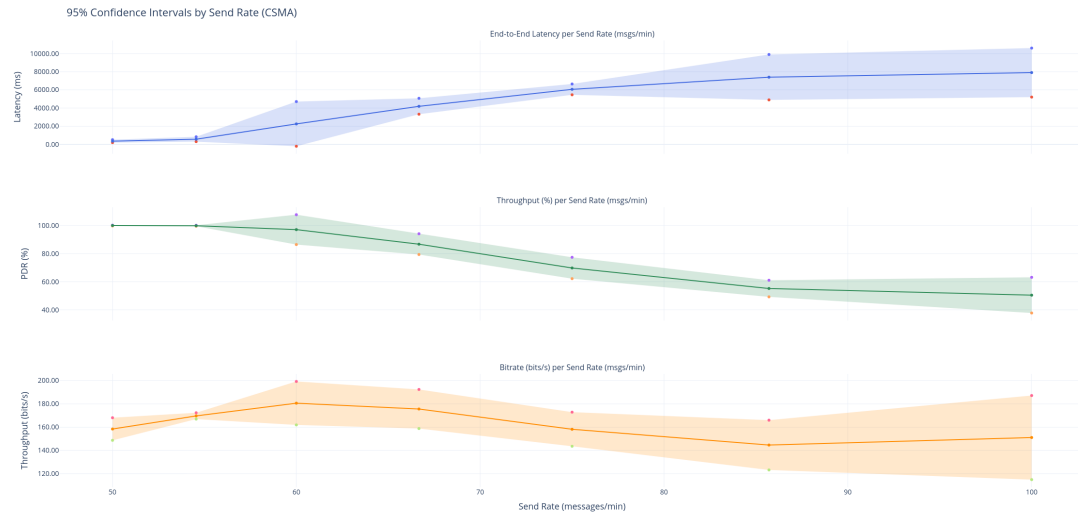


Figure 11: Impact of node movement on CSMA performance across different message rates. Metrics shown include packet delivery ratio, latency, and throughput.

## 5.5 Extended Disturbance Experiment: Moving Multiple Senders

To further investigate the effects of disturbances on protocol behavior, we performed an additional set of experiments involving a more aggressive topology change. In this scenario, we **moved Nodes 6, 25, and 26**, which were originally placed near the receiver—to **more distant locations after 300 seconds** of simulation time. This was done once the network had stabilized and routing paths were established. After the movement, **Node 3 remained the only sender close to the receiver**.



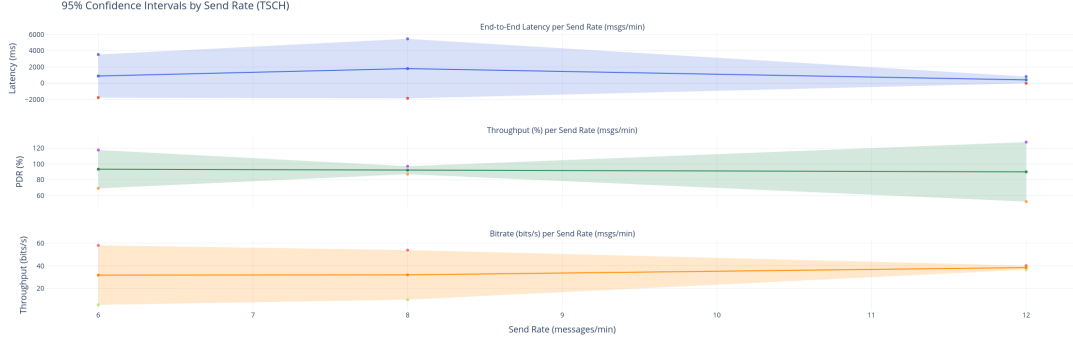


Figure 12: Impact of node movement on TSCH with Orchestra performance across different message rates. Metrics shown include packet delivery ratio, latency, throughput.

The intention was to create a more impactful disturbance, affecting both the topology and the traffic concentration near the sink. We aimed to observe how CSMA and TSCH with Orchestra react to such a scenario, especially in terms of routing adaptation, congestion, and overall delivery performance.

### 5.5.1 Observations and Interpretation

In this experiment, we clearly saw that both CSMA and TSCH with Orchestra performed worse after moving Nodes 6, 25, and 26 farther away from the receiver. We can see those in Figure 13 and Figure 14.

For CSMA, the impact was especially severe. After the movement, the protocol struggled to maintain a working network, and in most cases, the receiver didn't get any messages at all from the relocated senders. This shows that CSMA has difficulty recovering from changes in the network layout, especially when key nodes are moved farther from the sink.

TSCH with Orchestra handled the situation better, but it was still affected. The number of received messages dropped, and delays became noticeably longer. Even though the network stayed partially functional, the Packet Delivery Ratio (PDR) dropped significantly, ranging between 15 % and 45%, depending on the sender. This shows that TSCH was more resilient than CSMA, but still struggled to adapt fully to the new topology.

In summary, both protocols were negatively affected by the node movement. CSMA had major problems maintaining connectivity, while TSCH continued to deliver some data, but with reduced performance and higher latency.

## 6 Exponential Backoff

Exponential backoff is a method used in wireless networks where devices share the same communication channel. When two or more devices try to send data at the same time, a collision can happen. To avoid this from repeating, each device waits for a random amount of time before trying again. If it fails again, the device waits even longer the next time. The wait time grows exponentially after each failed attempt. This helps reduce the chances of another collision and keeps the network from getting too congested ("Back-off Algorithm for CSMA/CD", 2024).

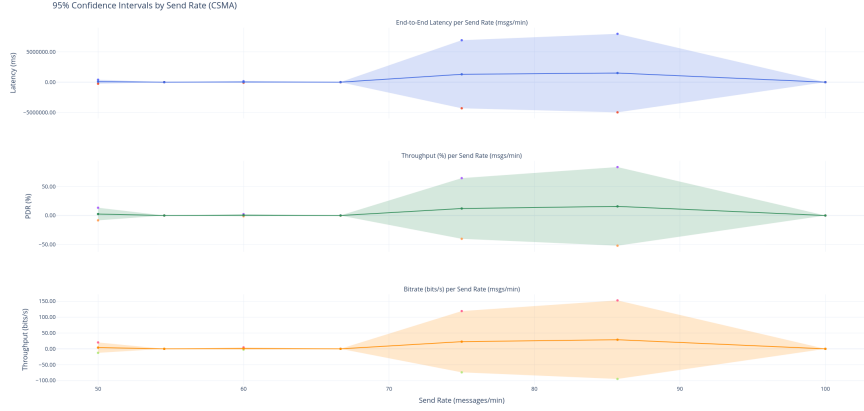


Figure 13: Performance of CSMA under extended disturbance (three sender nodes moved). Metrics include packet delivery ratio, latency, and throughput.

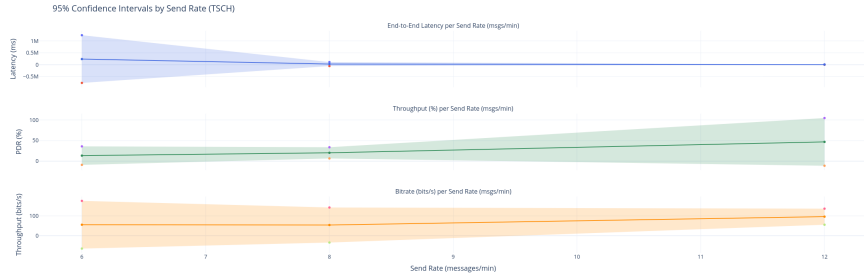


Figure 14: Performance of TSCH with Orchestra under extended disturbance (three sender nodes moved). Metrics include packet delivery ratio, latency, and throughput.

## 6.1 Exponential Backoff in CSMA-CA

In Carrier Sense Multiple Access with Collision Avoidance (CSMA-CA), such as in IEEE 802.11 (Wi-Fi), exponential backoff plays a role in the way devices share the communication channel (“IEEE 802.11 Standard”, 1997 and subsequent amendments; Tarek K. Refaat, 2013). When a device wants to send data, it first listens to the channel. If the channel is free for a short period, called DIFS (Distributed Interframe Space), the device sends its data. But if the channel is busy, the device waits until it becomes free, then starts a backoff process.

In this process, the device picks a random number of time slots to wait before trying to send. This number is chosen from a range that depends on the backoff exponent ( $BE$ ), which starts at a minimum value. The size of the range is:

$$\text{Backoff slots} \in [0, 2^{BE} - 1]$$

For example, if  $BE = 3$ , the device chooses a random number between 0 and 7. It then waits for that number of idle time slots. If the channel stays free, it sends the message. If no acknowledgment (ACK) is received, meaning a collision likely happened. The  $BE$  is increased by 1 (up to a maximum), and the process repeats with a larger backoff window.

This exponential growth in wait time helps avoid repeated collisions when many devices

want to send at the same time (Boggs et al., 1998; Stallings, Various Editions; Tanenbaum & Wetherall, Various Editions).

## 6.2 Backoff Mechanisms in TSCH

Time Slotted Channel Hopping (TSCH), used in IEEE 802.15.4e networks, works differently. In TSCH, devices follow a shared schedule where each one knows when to send and receive. Since transmissions happen at scheduled times, there's usually no need to compete for the channel ("IEEE 802.15.4e Standard", 2012; Yousaf et al., 2020). This reduces the chance of collisions and makes communication more predictable (Vasseur et al., 2015).

However, TSCH also supports shared time slots, where more than one device may try to send at the same time. In these shared cells, a backoff mechanism similar to CSMA-CA is used to avoid collisions. When two or more devices try to use the same slot, they use a random delay before retrying. Although the exact algorithm may be different, the goal is the same: to avoid repeated collisions by spacing out retries ("Back-off Algorithm for CSMA/CD", 2024; "TSCH and 6TiSCH - Contiki-NG Documentation", n.d.; Vasseur et al., 2015).

## 6.3 Impact on Performance

The exponential backoff approach in CSMA-CA affects performance in several ways ("IEEE 802.11 Standard", 1997 and subsequent amendments; Stallings, Various Editions; Tanenbaum & Wetherall, Various Editions; Tarek K. Refaat, 2013):

### Positive Effects:

- **Fewer Collisions:** Increasing the backoff time after each failed attempt lowers the chance that two devices will try to send at the same time again (“Back-off Algorithm for CSMA/CD”, 2024).
- **More Fairness:** Random wait times help ensure that all devices get a fair chance to send, especially over time.

### Negative Effects:

- **More Delay:** As the backoff time grows after each collision, it can take longer for a device to send its data, especially in crowded networks.
- **Lower Throughput:** While devices are waiting, the channel isn’t being used, which can waste bandwidth.
- **Not getting a chance to send:** In very busy situations, a device might have to wait too long or keep retrying without success.

TSCH, thanks to its schedule-based approach, avoids most of these problems. It offers consistent performance even when many devices are active. Still, in shared time slots where devices can collide, a backoff system is necessary. If many devices use the same shared slot, the backoff delays can increase, causing some of the same problems seen in CSMA-CA. So, overall performance depends on how well scheduled and shared slots are balanced in the network. (“IEEE 802.15.4e Standard”, 2012; “TSCH and 6TiSCH - Contiki-NG Documentation”, n.d.; Vasseur et al., 2015).

## 7 Impact of Slotframe Size on Performance

The *slotframe* is a component of the TSCH (Time Slotted Channel Hopping) MAC layer. It defines a cycle of time slots that repeats periodically and serves as the basic unit for organizing communication in the network. Each time slot within the slotframe can be assigned to a specific pair of transmitting and receiving nodes, either as a dedicated link or a shared cell. Nodes use this predefined schedule to determine when to wake up, transmit, receive, or sleep, which enables energy-efficient and collision-free communication.

The size of the slotframe directly affects the performance such as latency, throughput, and energy consumption (Palattella et al., 2013; Saleem et al., 2014). A **shorter slotframe size** allows nodes to access their allocated slots more frequently, which can lead to **lower latency** (Saleem et al., 2014). However, shorter slotframes have fewer time slots in each cycle, which means less chance for devices to send data. This can cause more competition for the channel if devices share slots, and may reduce the total amount of data that can be sent (Raza et al., 2012).

On the other hand, a **larger slotframe size** increases the time between slot repetitions for individual nodes, resulting in **higher latency** (Saleem et al., 2014). On the other hand, larger slotframes have more time slots for sending data, which works better when there are many devices. They also let devices sleep longer between sending, which helps save battery power (Palattella et al., 2013).

The **optimization of slotframe size** according to traffic patterns and application requirements is essential for achieving a balance between throughput, latency, and energy usage in TSCH deployments (Pereira & Silva, 2020).

## 8 Bandwidth, Packet Size, and Channel Usage in TSCH vs CSMA

Both TSCH and Unslotted CSMA-CA operate on the IEEE 802.15.4 standard, which provides a physical layer data rate of **250 kbps** in the 2.4 GHz band. However, the actual throughput achievable by each protocol differs significantly due to how the medium is accessed and how channels are utilized.

**CSMA** relies on a single channel and contention-based access using carrier sensing. While it performs well in sparse networks, its efficiency drops significantly in dense or high-traffic environments due to collisions, backoffs, and the hidden terminal problem. As more nodes attempt to transmit, contention increases, and the effective throughput often drops below **50–80 kbps**. In our testcase Since all nodes share the same channel, interference and congestion accumulate, limiting scalability.

**TSCH (Time Slotted Channel Hopping)**, by contrast, uses **multiple frequency channels** in combination with time slots. Each transmission occurs in a specific time slot on a pseudo-random channel chosen from a hopping sequence. This approach brings several advantages:

- **Frequency diversity:** mitigates the impact of narrowband interference and multipath fading.
- **Collision avoidance:** coordinated scheduling prevents simultaneous transmissions on the same time slot and channel.
- **Parallelism:** multiple node pairs can communicate at the same time as long as their slots and channels do not overlap.

The result is that TSCH can maintain higher throughput and lower collision rates in dense or noisy environments. With optimal scheduling and enough available slots, TSCH can approach an effective throughput of up to **100 kbps** per node pair. In contrast, CSMA becomes saturated quickly as it cannot spatially or temporally separate traffic.

Regarding packet size, both protocols inherit the IEEE 802.15.4 MAC frame limit of **127 bytes**, which includes headers and optional security overhead. This leaves an application-level payload of roughly **80–100 bytes** per packet in most practical cases.

### 8.0.1 Example Per-Node Throughput Calculation for CSMA

As an example we take the Throughput graph of CSMA, see Figure 9. At 60 msgs/minute the measured throughput is 169.97 bits/second per senders node. With a payload of 25 bytes (200 bits), the ideal throughput per node is 200 bits/s. The measured throughput is 169.97 bits/s per node, resulting in an effective delivery ratio of:

$$\frac{169,97}{200} \approx 84.99\%$$

### 8.0.2 Packet Delivery Ratio (PDR) Measurement in TSCH

To evaluate the reliability of the TSCH Orchestra protocol, we calculate the **Packet Delivery Ratio (PDR)** as the percentage of received messages at the sink node relative to the number of messages originally **sent** by the sender nodes:

$$\text{PDR} = \frac{\text{Received Messages}}{\text{Sent Messages}} \times 100\%$$

This approach differs from less accurate methods that use only *confirmed* TSCH transmissions as the denominator, which may overlook messages that were sent but not acknowledged in time, especially under congestion. By using the total sent messages as the reference, our measurement reflects the true end-to-end reliability from sender to receiver.

In theory, at a send rate of 6 messages per minute per node (20 nodes total), each node transmits 25-byte messages, resulting in a maximum throughput of 20 bits/second. Our measured throughput reaches approximately 18.96 bits/second, indicating a **theoretical PDR of 94.8%**.

However, the **actual PDR** graph in Figure 4 shows a lower maximum of around **80%**. This discrepancy arises from the fact that we compute the **PDR relative to all messages sent**, not only those confirmed by the MAC layer. As such, our results offer a more stringent and realistic view of network reliability under TSCH.

**Conclusion: TSCH Orchestra Has a Higher Theoretical PDR Than CSMA** Our measurements of actual throughput versus theoretical expectations confirm that TSCH, by design, offers a higher theoretical PDR than CSMA due to its collision-free, time-slotted communication and frequency hopping. However, in our high-density WSN scenario, the practical PDR of TSCH Orchestra drops significantly. This is caused by queue saturation at critical relay nodes, which become bottlenecks in the routing paths to the sink. As a result, messages are dropped before transmission, despite TSCH’s scheduling advantages.

Feature	CSMA (Single Channel)	TSCH (Multi-Channel)
Channel usage	1 static channel	Multiple hopping channels
Collision handling	Backoff and retry	Avoided by schedule
Interference robustness	Low	High (via frequency diversity)
Scalability in dense networks	Poor	Good
Typical effective throughput	< 80 kbps	Up to 100 kbps
Max MAC frame size	127 bytes	127 bytes
Usable payload	~80–100 bytes	~80–100 bytes

Table 3: Comparison of CSMA and TSCH in terms of channel usage and throughput

## 9 Future academic work

### 9.1 TSCH Synchronization lost at Low Traffic

In Figure 4 the TSCH PDR curve starts around 70–80% at 3 msgs/min, then slightly dips before increasing again around 4–6 msgs/min, and only then begins the expected decline as the traffic increases. This subtle dip is counterintuitive—we normally expect high PDR at low loads.

As written in Hunde et al., 2021 TSCH networks depend on time synchronization, maintained via periodic message exchanges. Nodes rely on traffic from parents and the PAN coordinator to adjust their clocks. At extremely low send rates, insufficient packet exchange may lead to desynchronization, causing unexpected packet loss and a drop in PDR. This issue does not exist in CSMA, which operates asynchronously, and was not discussed in Lauwens et al.’s analysis.

While 3-4 msgs/min does not seem extremely low, we can clearly observe the dip in the PDR plot. This could be subject for further research.

## 9.2 Latency Variance and PDR Recovery

An interesting phenomenon observed during testing is that, at specific message rates (e.g., 8, 12, 30 msgs/min), the latency variance in TSCH networks increases sharply while the PDR begins to recover after a drop. This suggests a correlation between increased timing variability and successful delivery adaptation. The interplay between synchronization recovery, scheduling adaptation, and MAC-layer variance is probably unexplored in literature, offering a promising direction for future academic work on reliability-aware TSCH scheduling.

## 10 APPENDIX

**Use of ChatGPT in This Project** During this project, OpenAI's ChatGPT was used as a support tool for generating code snippets, analyzing parts of the collected data, and improving the clarity and structure of written sections. Its assistance was limited to accelerating development and enhancing textual quality; all results and interpretations were verified and validated independently.

**Random sending pattern** With script `deltaSendings.py` which measures the delta between 2 messages that are send, we can prove the random sending pattern

602.10s - 'Msg fd00::206:6:6:6 0' (first message)  
 625.56s - 'Msg fd00::206:6:6:6 1' (delta 23.46s)  
 639.92s - 'Msg fd00::206:6:6:6 2' (delta 14.36s)  
 ...  
 2212.80s - 'Msg fd00::206:6:6:6 99' (delta 23.43s)  
 Number of messages send by node 6: 100

**Github respository** All code is available on Github via [https://github.com/mdequanter/2024\\_2025\\_Project\\_MPA.git](https://github.com/mdequanter/2024_2025_Project_MPA.git)

**Extra project, used for research** 5GWebRtc – Encrypted Image Streaming Testbench over WebSocket, code available via <https://github.com/mdequanter/5GWebRtc.git>

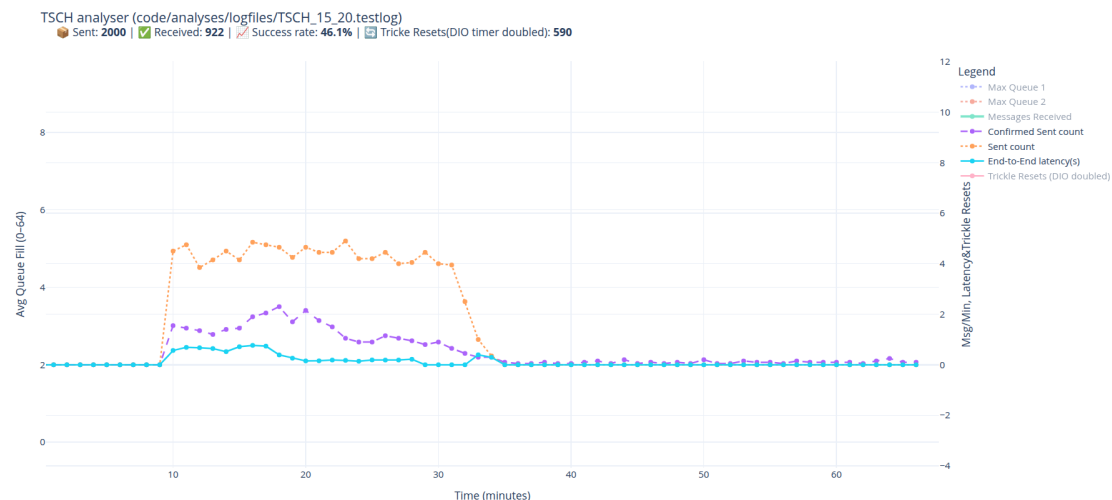


Figure 15: Plotly possibilities to search for specific data

### Plotly possibilities to search for specific data

**Confidence intervals based on testbins, not messages/minutes** You can see the individual confidence intervals for each sendrate bins as mentioned in Table 2. A ratio of **sendrate X CLOCK\_SECONDS** is used in the **sender-node.c**. (see Figure. 16 and 17)



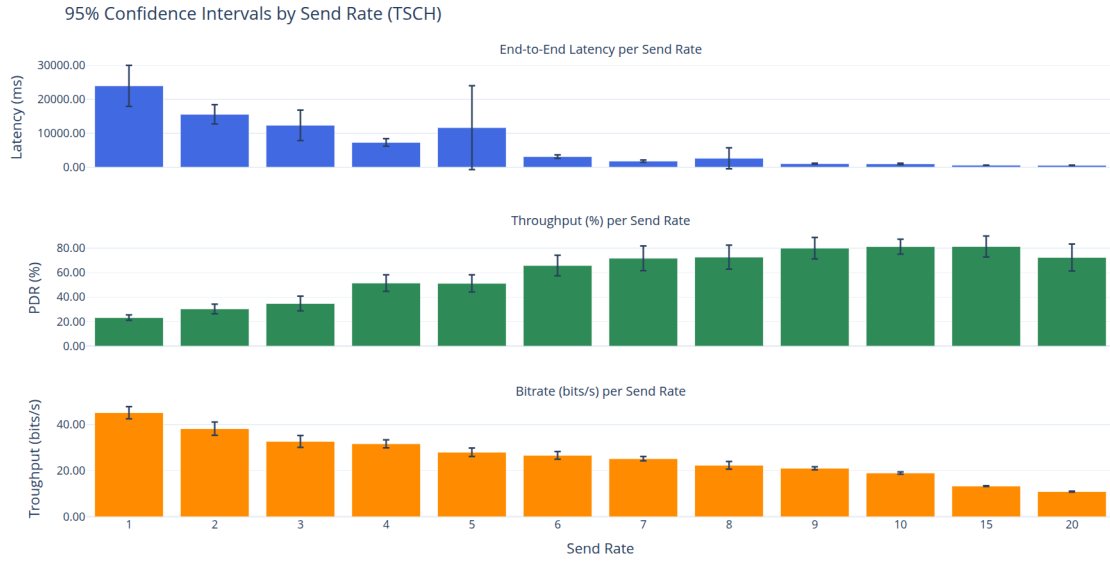
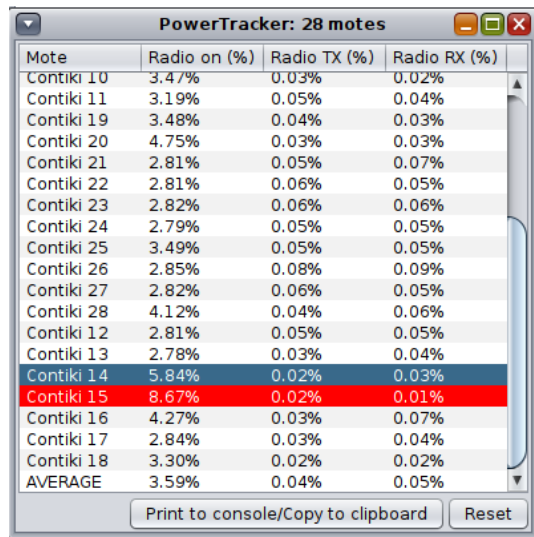


Figure 16: 95 Confidence intervals TSCH per send rate bin



Figure 17: 95 Confidence intervals CSMA per send rate bin

**TSCH vs CSMA radio On** As you can see on Figure 19 and 18 during the tests CSMA used 100% Radio, while TSCH Orchestra only 3,59% on average.

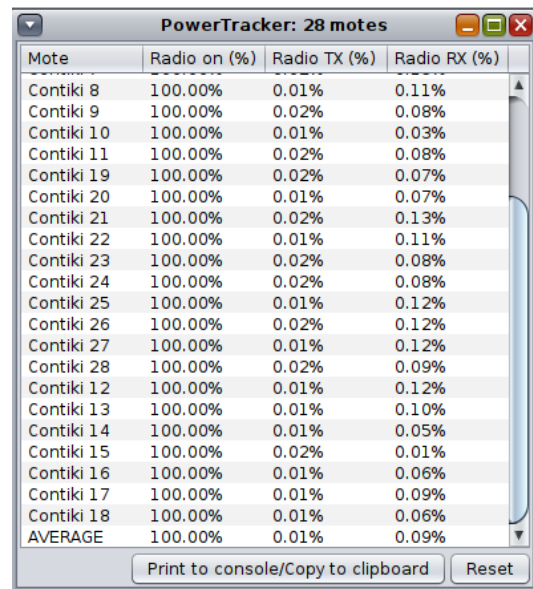


PowerTracker: 28 motes

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Contiki 10	3.47%	0.03%	0.02%
Contiki 11	3.19%	0.05%	0.04%
Contiki 19	3.48%	0.04%	0.03%
Contiki 20	4.75%	0.03%	0.03%
Contiki 21	2.81%	0.05%	0.07%
Contiki 22	2.81%	0.06%	0.05%
Contiki 23	2.82%	0.06%	0.06%
Contiki 24	2.79%	0.05%	0.05%
Contiki 25	3.49%	0.05%	0.05%
Contiki 26	2.85%	0.08%	0.09%
Contiki 27	2.82%	0.06%	0.05%
Contiki 28	4.12%	0.04%	0.06%
Contiki 12	2.81%	0.05%	0.05%
Contiki 13	2.78%	0.03%	0.04%
Contiki 14	5.84%	0.02%	0.03%
Contiki 15	8.67%	0.02%	0.01%
Contiki 16	4.27%	0.03%	0.07%
Contiki 17	2.84%	0.03%	0.04%
Contiki 18	3.30%	0.02%	0.02%
AVERAGE	3.59%	0.04%	0.05%

Print to console/Copy to clipboard Reset

Figure 18: TSCH radio ON while performing tests



PowerTracker: 28 motes

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Contiki 8	100.00%	0.01%	0.11%
Contiki 9	100.00%	0.02%	0.08%
Contiki 10	100.00%	0.01%	0.03%
Contiki 11	100.00%	0.02%	0.08%
Contiki 19	100.00%	0.02%	0.07%
Contiki 20	100.00%	0.01%	0.07%
Contiki 21	100.00%	0.02%	0.13%
Contiki 22	100.00%	0.01%	0.11%
Contiki 23	100.00%	0.02%	0.08%
Contiki 24	100.00%	0.02%	0.08%
Contiki 25	100.00%	0.01%	0.12%
Contiki 26	100.00%	0.02%	0.12%
Contiki 27	100.00%	0.01%	0.12%
Contiki 28	100.00%	0.02%	0.09%
Contiki 12	100.00%	0.01%	0.12%
Contiki 13	100.00%	0.01%	0.10%
Contiki 14	100.00%	0.01%	0.05%
Contiki 15	100.00%	0.02%	0.01%
Contiki 16	100.00%	0.01%	0.06%
Contiki 17	100.00%	0.01%	0.09%
Contiki 18	100.00%	0.01%	0.06%
AVERAGE	100.00%	0.01%	0.09%

Print to console/Copy to clipboard Reset

Figure 19: CSMA radio ON while performing tests

**Team Members' Work Contributions** During the class sessions, Maarten and Imad collaborated closely. The test scenarios were discussed, decided upon, and set up together in Cooja. The first tests were executed collaboratively.

Maarten was primarily responsible for programming, setting up the automated test bins, and developing the analytics scripts used to generate graphs for latency, PDR, and throughput.

Throughout the final weeks, both team members regularly discussed intermediate results to improve the quality and reliability of the project.

In the final week, both members contributed to writing the report. Specific sections were handled as follows:

- **Maarten:** Introduction, Hypothesis and Reasoning, Experimental Setup, Results and Discoveries, Bandwidth, Packet Size, and Channel Usage in TSCH vs CSMA, Future academic work, Appendix
- **Imad:** Behavior Under Disturbances, Exponential Backoff, Impact of Slotframe Size on Performance

The presentation will be delivered jointly.

## References

- Back-off algorithm for csma/cd [Provides a clear explanation of the backoff algorithm, though for CSMA/CD, the principles are similar]. (2024). *GeeksforGeeks*. <https://www.geeksforgeeks.org/back-off-algorithm-csmacd/>
- Boggs, D., Mogul, J., & Kent, C. (1998). A proposed standard for ethernet (ieee 802.3) [While focused on wired Ethernet, it details the principles of binary exponential backoff.]. *RFC*, (2280). <https://datatracker.ietf.org/doc/html/rfc2280>
- Hunde, E., Deac, D., Thielemans, S., Carlier, M., Steenhaut, K., Braeken, A., & Dobrota, V. (2021). Time slotted channel hopping and contikimac for ipv6 multicast-enabled wireless sensor networks. *Sensors*, *21* (5), 1771. <https://doi.org/10.3390/s21051771>
- Ieee 802.11 standard [The foundational document defining Wi-Fi and CSMA-CA. Refer to sections on DCF and BEB.]. (1997 and subsequent amendments). *IEEE*.
- Ieee 802.15.4e standard [Defines TSCH. Focus on sections describing time slot management and channel hopping.]. (2012). *IEEE*.
- Lauwens, B., Scheers, B., & Van de Capelle, A. (2010). Performance analysis of unslotted csma/ca in wireless networks. *Telecommunication Systems*, *44*, 109–123.
- Palattella, M. R., Dohler, M., Grieco, L. A., Rizzo, A., Torsner, J., & Engelke, P. (2013). On the energy consumption of ieee 802.15.4e tsch networks. *Sensors*, *13*(10), 13571–13596.
- Pereira, J., & Silva, J. P. a. (2020). Slotframe optimization for ieee 802.15.4e tsch networks with traffic differentiation, 7–14.
- Raza, S., Misic, J., & Misic, V. B. (2012). Performance evaluation of ieee 802.15.4e tsch under different network densities and traffic loads. *arXiv preprint arXiv:1207.1877*.
- Saleem, M., Jan, M. Z., Farman, A., & Asghar, M. B. (2014). Delay analysis of ieee 802.15.4e time slotted channel hopping (tsch) mechanism. *International Journal of Distributed Sensor Networks*, *2014*.
- Shah, G. A., & Akan, O. B. (2014). Performance analysis of csma-based opportunistic medium access protocol in cognitive radio sensor networks. *Ad Hoc Networks*, *15*, 4–13.
- Stallings, W. (Various Editions). Wireless communications & networks [Covers CSMA-CA extensively, including the backoff mechanism.]. *Pearson Education*.
- Tanenbaum, A. S., & Wetherall, D. J. (Various Editions). Computer networks [Provides a detailed explanation of CSMA-CA and the binary exponential backoff algorithm.]. *Pearson Education*.
- Tarek K. Refaat, H. H. A., Ramez M. Daoud. (2013). Ieee 802.11 standards. [Provides an overview of IEEE 802.11 standards.]. *Intelligent Control and Automation*, *4* (4). <https://www.scirp.org/reference/referencespapers?referenceid=1000353>
- Tsch and 6tisch - contiki-ng documentation [Explore the documentation related to Contiki-NG's TSCH implementation.]. (n.d.). *Contiki-NG*. <https://docs.contiki-ng.org/en/develop/doc/programming/TSCH-and-6TiSCH.html>
- Vasseur, J., Vilajosana, X., & Puiatti, A. (2015). Using ieee 802.15.4e time-slotted channel hopping (tsch) in the internet of things (iot): Problem statement [Explains the design principles of TSCH and its collision avoidance through scheduling and mentions backoff for shared cells.]. *RFC*, (7554). <https://datatracker.ietf.org/doc/html/rfc7554>
- Yousaf, M., Aslam, M., Ahmad, H., Javaid, N., & Khan, S. (2020). A reliable data transmission model for ieee 802.15.4e enabled wireless sensor network under wifi interference [Details the MAC sublayer specifications for IEEE 802.15.4e]. *MDPI*, *17*(6). <https://www.mdpi.com/1424-8220/17/6/1320>