

Project Report

Sensor data collection & data aggregation

07.05.2020

Members:

Vladimír Ročín, AU ID: 646268

Martin Tomko, AU ID: 644417

Tomáš Samuel Lang, AU ID: 647056

Denis Hlinka, AU ID: 647055

Wireless Sensor Networks

Department of Engineering

Aarhus University

Glossary	3
Introduction	4
Problem Analysis	5
Data Flow concept	6
Experiment design	7
5.1. Environment	7
5.2. Networking	8
5.3. Node design	8
5.3.1. Sensor node	8
5.3.2. Aggregator node	9
5.3.3. Sink node	10
5.4. Security	11
Results	11
Conclusion	15
Course gain	15

1. Glossary

OS - Operating System

CFS - Coffee file system

DOS - Denial of service

UDP - User datagram protocol

IP - Internet protocol

DODAG - Destination-Oriented Directed Acyclic Graph

RX - Reception/Receiving

TX - Transmitting

WSN - Wireless sensor networks

2. Introduction

The assigned mini-project for this group was project 2 with name “Sensor data collection and aggregation”. The goal of this project was to set up a simple multi-hop sensor network, perform data collection on sensor nodes and periodically send those events and data to a sink which works as a data-collector node. The setup should run in two modes: with and without data aggregation. The results of both will be compared in terms of accuracy and power consumption.

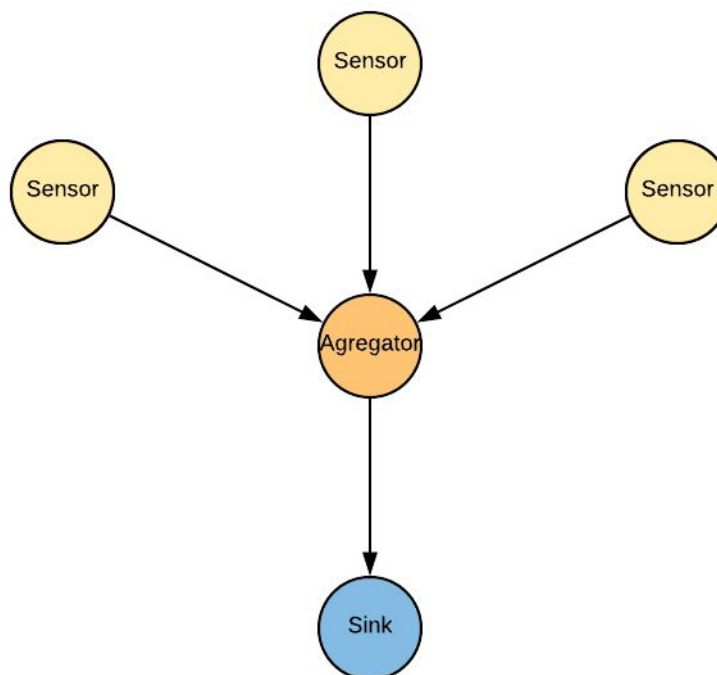


Figure 1: Aggregation principle

In Figure 1 is described the principle of aggregation. Multiple sensor nodes are sending data to the aggregator node which makes aggregation operations on these data (Average for example) and sends aggregated data to the sink. The result of such an aggregation is then reduced data traffic to the sink node and reduced power consumption for data transmission.

3. Problem Analysis

The question is, how can aggregation reduce traffic in our networks, how the power consumption can be reduced and what would be consequences of aggregation. Multiple experiments were made to clarify all of those questions.

Firstly, aggregation should be explained. By aggregation is meant reduction of the amount of data that flows through the network. There are many aggregation operations like max, min, average, sum. Network topology design is also very important when it comes to aggregation because in some topologies it is pointless to use data aggregation.

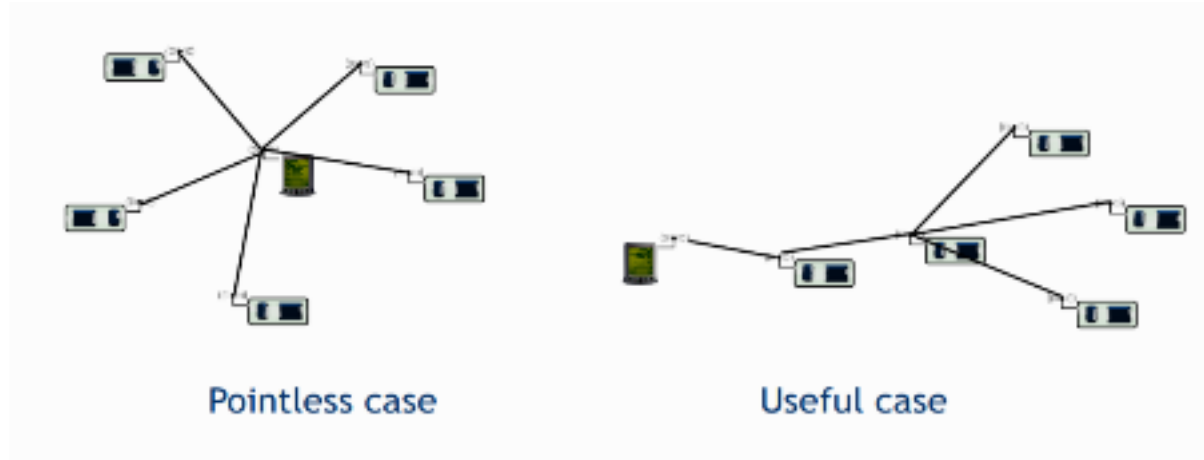


Figure 2: pointless and useful aggregation

In Figure 2 we can see 2 types of topologies. Star topology where all nodes communicate with sink (server) directly. In such a case making aggregation doesn't make much sense since each node would aggregate only its own data and they don't receive any data from other nodes where the aggregation could be performed. On the other side of figure 2, we can see a node that receives the communication from 3 other nodes. This node forwards the communication to the server through another node on the left side. In this case nodes that are relaying others can actually store received values, perform aggregation once there is enough data and then send these aggregated values to their next hop. This results in reduction of data transmission since only one message with aggregated data will be sent instead of N messages.

4. Data Flow concept

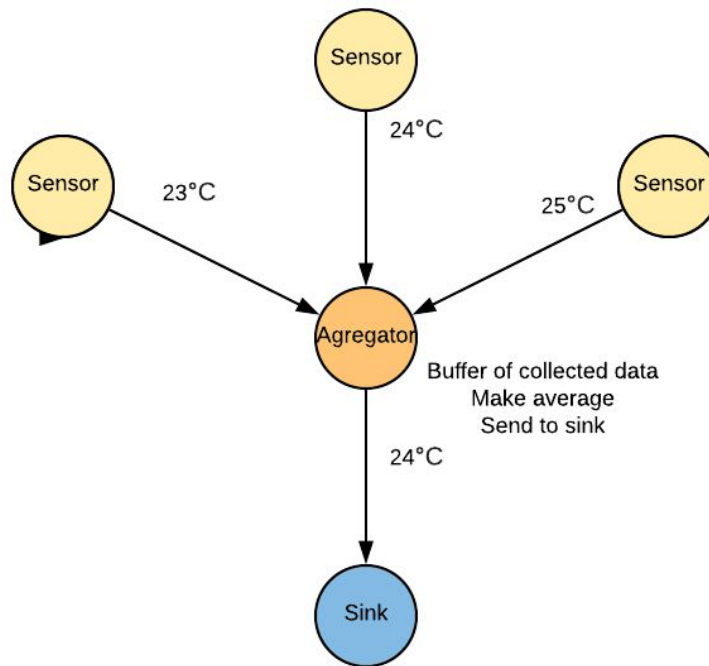


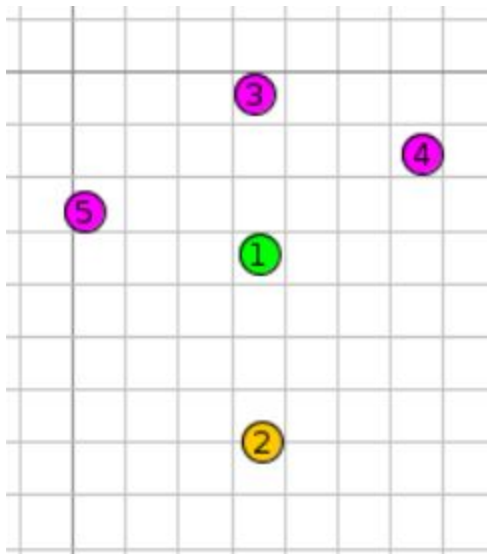
Figure 3: Data flow concept

Two types of aggregation will be implemented in the project. Temporal aggregation on the Sensor node where the sensor itself will have a configurable buffer to collect data when the buffer is full, average values are sent to the aggregator. Second aggregation will be executed on the aggregator node. Aggregator also contains a buffer where it collects data from sensors and as with the simple aggregation when the buffer is full, average is executed and aggregated data will be sent to the sink node to store them. Due to data aggregation accuracy can be lost since only average values of N measurements are sent. When it comes to temperature data we need to know instantly when temperature exceeds threshold values and some action is required. In this situation, the Aggregator detects that the value exceeds the set threshold and this value will be immediately sent to the sink. In this case the system will not wait until the buffer is full.

5. Experiment design

5.1. Environment

For experiment implementation, Cooja simulators with contiki-NG OS were used. Contiki-OS is a lightweight operating system for network nodes and Cooja simulator can simulate nodes and their networking without actual physical devices. Cooja simulator is running on linux devices so virtual machines were used.



In Figure 4 is shown the topology of nodes in the Cooja simulator. Same topology as mentioned and discussed in concept ideas with 3 types of nodes was used. Node 1 (Green) which represents the aggregator and also the root of this small network. Node 2 (Orange) which represents sink (data collector) and sensor nodes 3,4,5 (Purple).

Figure 4: Experiment topology

Cooja simulator gives access to logs of all nodes (see Figure 5) together with their simulator interfaces such as Buttons, CFS and others.

Time	Mote	Message
4:22:50.754	ID:2	[INFO: SINK] Data received: 1786 - 21nk
4:22:51.063	ID:1	[INFO: AGREGATOR] Storing received data 0 - '12nk'
4:22:51.259	ID:1	[INFO: AGREGATOR] Storing received data 1 - '29nk'
4:22:51.264	ID:1	[INFO: AGREGATOR] EXTREME DETECTED!! Prioritizing message!
4:22:51.325	ID:2	[INFO: SINK] Data received: 1787 - 29nk
4:22:55.723	ID:1	[INFO: AGREGATOR] Storing received data 2 - '29nk'
4:22:55.727	ID:1	[INFO: AGREGATOR] EXTREME DETECTED!! Prioritizing message!
4:22:55.770	ID:2	[INFO: SINK] Data received: 1788 - 29nk

Figure 5: Logs example

5.2. Networking

Contiki-NG allows UDP communication using RPL protocol with IPv6. RPL is routing protocol for lossy and low power networks that builds and maintains DODAG topology. RPL supports any direction traffic such as node to root, root to nodes or node to node. RPL handles also removing nodes and adding new nodes to the topology on the go while still maintaining multi-hop communication when possible. For communication with the internet we are using a border node which is also root. Using node's serial-io server we tunneled communication using linux tool "tunslip6". This was made mainly for sending data to cloud service with a website to interpret data on graphs. Project ended up only storing data into a local file on the sink node using CFS.

5.3. Node design

5.3.1. Sensor node

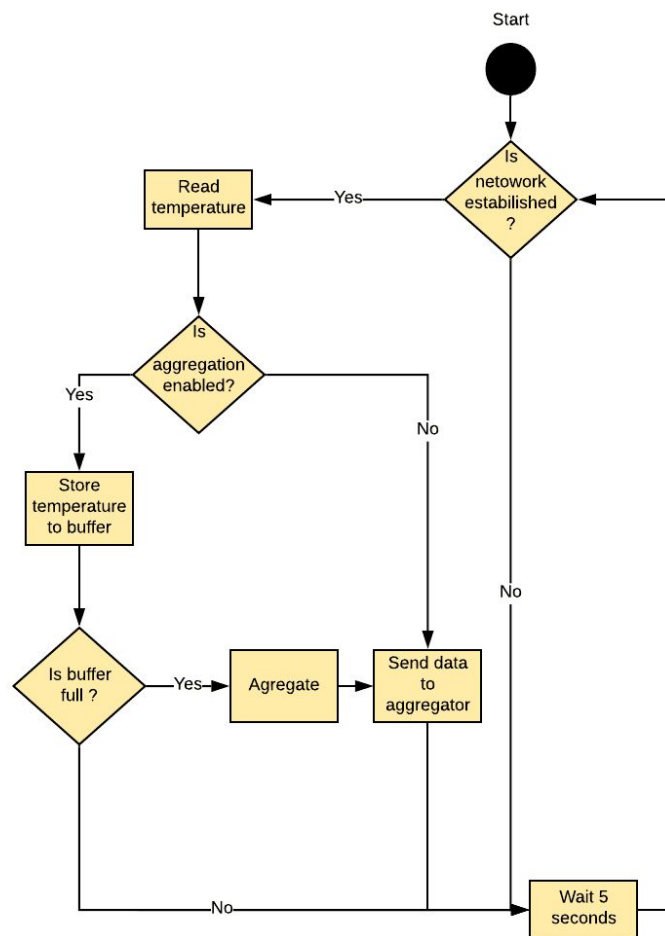


Figure 6: Sensor Activity diagram

Sensor node is responsible for data measures. It includes very simple aggregation by itself. This can be enabled or disabled by changing the boolean variable altogether with the size of the buffer which collects data for aggregation. For our experiments we decided to do a buffer of size 3. This number depends on the use of our readings. The more accurate data we require and faster we require them the size of this buffer should be smaller. We decided to wait 5 seconds between every data reading. This value is another thing to discuss. Value again depends on use of sensors, there is trade-off between accuracy and power consumption.

5.3.2. Aggregator node

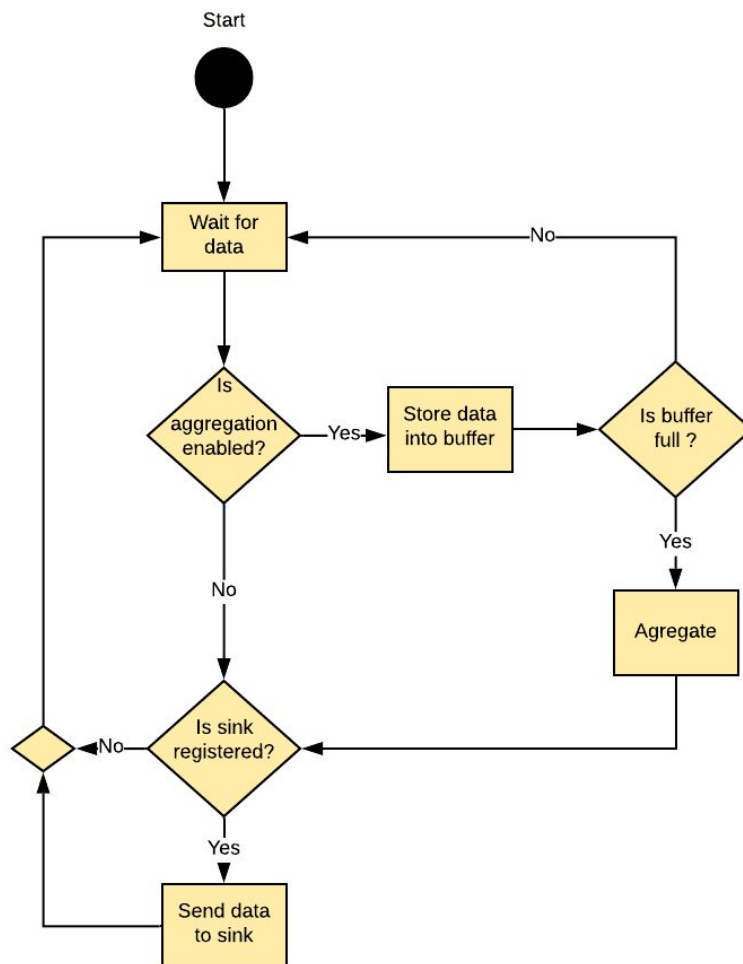


Figure 7: Aggregator activity diagram

Agregator node can work also in 2 modes. If aggregation is disabled, node functionality is reduced to work only as relay. Whatever data comes to this node, they are sent instantly to

the sink. While in the mode where aggregation is enabled, it stores data into buffer until buffer is full. Buffer size can be configured and as discussed above, depends on use of the system. Aggregation is based on the same principle as in sensor nodes, fill the buffer, make average and send data to sink. In Figure 7 we can see that the system is checking if the sink is registered. When the sink is not registered, the aggregator will not send any data anywhere. For sink registration a very simple communication protocol was created where aggregators check for registration messages. If Registration message is received, the sender's ip is stored into a variable as sink and sends a confirmation message back to sink. When a new sink registration message is received, sink address is overridden and a new sink will become an active one.

Agregator is also responsible for event detection. If the temperature measured by the sensor is too high or too low. Message is still stored and handled by aggregation, but this message is also instantly forwarded to the sink without waiting for the buffer to fill up. This ensures that all extreme values are sent to the sink in the shortest time possible.

5.3.3. Sink node

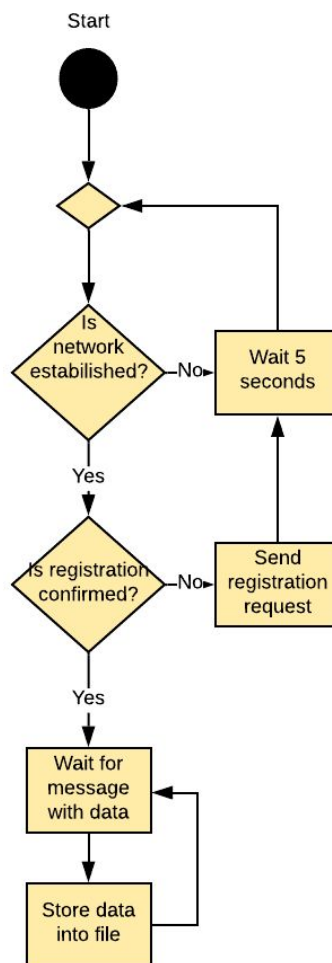


Figure 8: Sink activity diagram

Sink node works as a storage unit. When added to the network first it tries to register itself at aggregator nodes. It sends a registration message to the aggregator and then waits 5 seconds. If no confirmation message is received, another registration message is sent. After registration confirmation, sink actively waits for data and stores them into a text file. Nodes supports coffee filesystems for working with files. This registration protocol was made to ensure easy swapping between sinks. For example when some small storage device is attached to a node, and data should be retrieved. It's safe to put a new sink into the network, wait until its registration is completed and the old one can be safely turned off with the guarantee of not losing any data.

5.4. Security

Security wasn't the goal of this project but some of the security issues of this system were discussed. System is not protected by DOS or inserting false nodes to the network. Especially for this system hacker could put a modified sink into the network which would register itself to the aggregator and fully replace the original one accessing all the data. This problem could be fixed by adding some kind of password or authentication mechanism into sink registration. An aggregator would register a new sink only when authorization of the new sink is valid.

6. Results

We run the tests using 2 modes. First mode with aggregation disabled and then second time with aggregation enabled. Tests were performed also using different sizes of buffers (3,5,7). First thing to look at was the number and frequency of messages incoming to our sink. Tests were running in simulation for 15 minutes, Results can be seen in Figure 9.

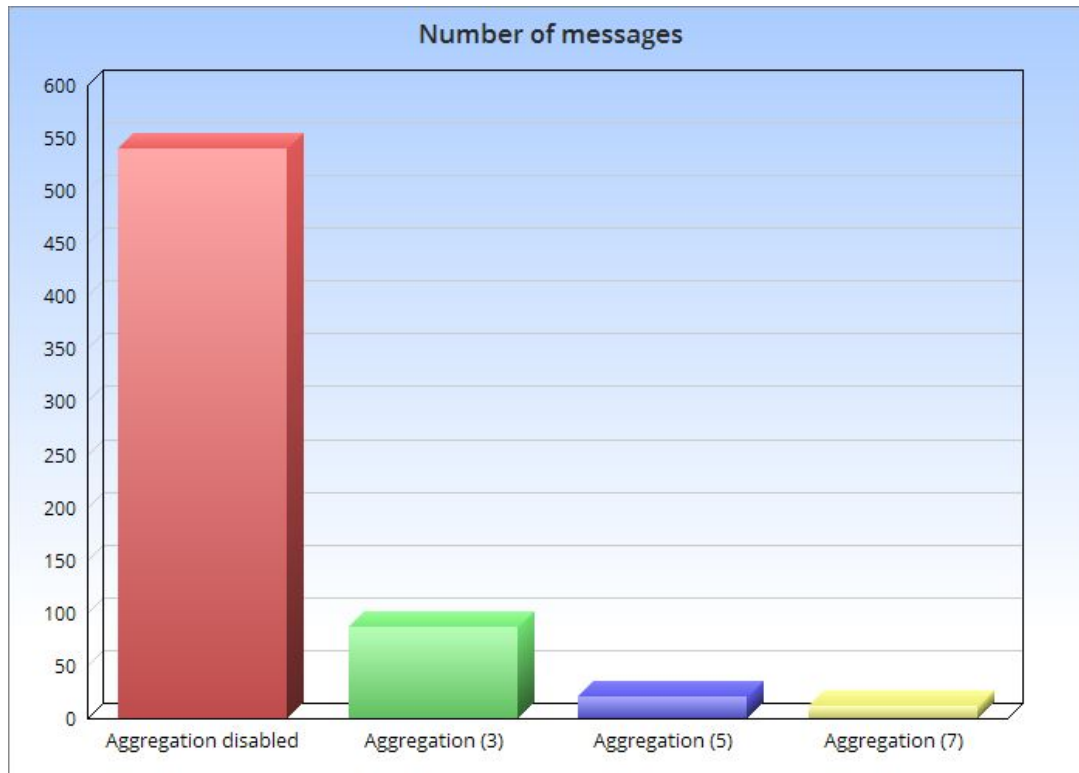
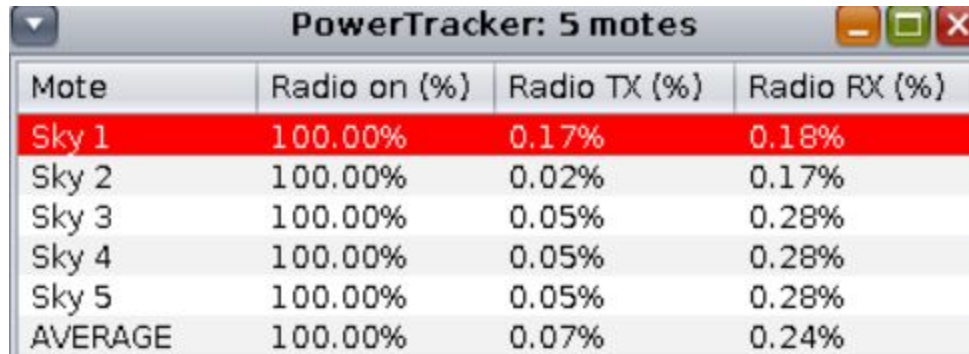


Figure 9: Numbers of messages on sink

Numbers of messages received on sink drastically decreased while using aggregation. Without aggregation it was around 540 messages in 15 minutes. Then using aggregation with a buffer size of 3, this value dropped to 86 messages per 15 minutes. For buffer size 5 it was 21 messages and with buffer size of 7 it was only 12 messages. Those values may vary on test rerun because of random event elements. When value is extreme, value is relayed directly to sink without waiting in buffer, and aggregation.

Next it is important to look at our radio transmitting. Transmitting consumes a lot of power and by aggregation we reduce the number of messages that have to be sent. Figure 10 shows how much time radio is transmitting and receiving data with disabled aggregation. First notable value is there that all nodes have the radio turned on the whole time. For better power consumption it would be better to turn it off after data transmit, and turn it on right before it is needed again.

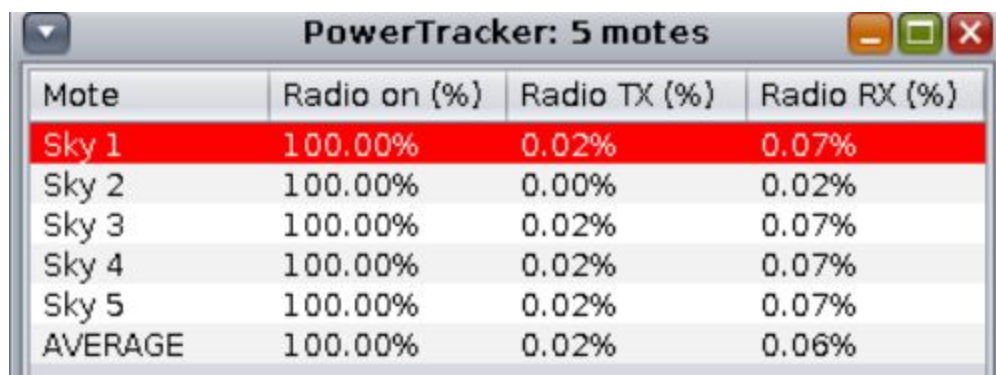


Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	100.00%	0.17%	0.18%
Sky 2	100.00%	0.02%	0.17%
Sky 3	100.00%	0.05%	0.28%
Sky 4	100.00%	0.05%	0.28%
Sky 5	100.00%	0.05%	0.28%
AVERAGE	100.00%	0.07%	0.24%

Figure 10: Radio cycle without aggregation

If we compare these results to Figure 11, which shows radio states with aggregation with buffer size of 3. We can see some huge differences on Radio RX values. Because of local aggregation on sensor nodes, sensors have much smaller transmitting activity. Because sensors are very close to each other they transmit data to each other instead of sending them to aggregators but because messages are not meant for them they are ignored. This is reason why sensor nodes has some receiving percentage even they dont process received data.

Difference between enabled and disabled aggregation is very noticeable. When we look at average TX and RX aggregation reduced Radio activity by almost 4 times. Which is a huge difference for power consumption. Mostly for our “Sky 2” mote which is Sink. From 0.17% RX it decreased to 0.02 so aggregation reduced this communication by almost 9 times.



Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	100.00%	0.02%	0.07%
Sky 2	100.00%	0.00%	0.02%
Sky 3	100.00%	0.02%	0.07%
Sky 4	100.00%	0.02%	0.07%
Sky 5	100.00%	0.02%	0.07%
AVERAGE	100.00%	0.02%	0.06%

Figure 11: Radio cycle with aggregation (3)

Figure 12 shows radio activity with aggregation buffer of 5. We can see that this again reduced transmitting and receiving activity 2 times. More data is stored to buffers and averaged. This will start reducing power for radio use but on the other side we are losing accuracy on our data measurement.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	100.00%	0.01%	0.05%
Sky 2	100.00%	0.00%	0.01%
Sky 3	100.00%	0.01%	0.04%
Sky 4	100.00%	0.01%	0.04%
Sky 5	100.00%	0.01%	0.04%
AVERAGE	100.00%	0.01%	0.04%

Figure 12: Radio cycle with aggregation (5)

Difference between buffer size 5 and buffer size 7 is not very different. There is still a very small amount of activity reduced. But a comparison test without aggregation and aggregation with a buffer size of 7 is enormous. Radio RX for “Sky 2” mote (Sink) reduced from 0.17% activity to 0.01%.

Mote	Radio on (%)	Radio TX (%)	Radio RX (%)
Sky 1	100.00%	0.01%	0.03%
Sky 2	100.00%	0.00%	0.01%
Sky 3	100.00%	0.01%	0.03%
Sky 4	100.00%	0.01%	0.03%
Sky 5	100.00%	0.01%	0.03%
AVERAGE	100.00%	0.01%	0.02%

Figure 13: Radio cycle with aggregation (7)

All tests executed in the Cooja emulator system were running for 15 minutes. Radio power tracker was reseted after connection was established to remove duty cycles for creating DODAG network. Test included 3 sensors, 1 aggregator and 1 sink node.

7. Conclusion

Based on our experiment, where tests were run under 4 configuration (Without aggregation, aggregation with a buffer of 3,5,7). It's very clear that aggregation of data has a huge impact on radio transmitting usage. Usage of a system like this depends. For example when we measure temperature in the sewers where small differences in temperature doesn't very matter and we need just the average of temperatures. This solution will reduce power consumption significantly. On the other side, when we are in an environment where temperature values are very important (Power plants for example) , we need to be very careful with aggregation. Even if we make prioritizing messages with extreme values, small changes in temperature wouldn't be visible straight away because of the buffer. Aggregation design in a very strict environment must be properly discussed and tested.

8. Course gain

During this course our team gained a lot of interesting information about wireless networking and how IoT sensors work in real life. Even with small difficulties to work with real "Telos-B" sensors because of unfortunate restrictions due to Covid-19 we could still simulate a fully working environment in a Cooja simulator. Team had struggled with some challenges that came together with project work. One of them was actual programming in language "C" which we are not very familiar with. Team was able to handle almost all challenges that came either with unfamiliar programming language or simulated environment for WSN. For the whole team it was very interesting to work with IoT sensors. It was different from usual programming challenges.