DESIGN REPORT

MaritimeManet: Distributed Neighborhood Discovery

Faculty of Electrical Engineering, Mathematics and Computer Science

Supervised by:

Dr.Ir. P.T. de Boer

Authors:

Vladimirs Popovs - s2310651

Aly Hanee Mohamed Samih Afia - s2087529

Omar Ahmed Mohamed Gadelmawla - s2371146

Lola Solovyeva - s2207958

CONTENTS

INTRODUCTION

Marine communication is a significant element to ensure the safety of the crew, passengers, and the cargo in emergency situations. It involves transmitting the information between ships and from ships to the land base. To lower the risk of accidents, there has to be a reliable and foolproof way for communication in any situation. This research paper provides an information about the idea that could change how the marine communication is performed. This report is going to discuss a new concept for transmitting the information on the sea and show the algorithm that is responsible for consistently discovering strongest possible connection between the ships. This chapter will make a reader acquainted with a Thales Group and show their domain and values. Further, it will briefly describe the idea to give the first insights. Lastly, it will show why marine communication still needs improvements and demonstrate a currently existing way of communication.

THALES

Thales Group is a global French multinational company that is focused on digital and “deep tech” innovations – artificial intelligence, Big Data, cybersecurity, and quantum technology. Their innovations and technologies are used worldwide in fields concerning aerospace, transportation, security, and digital identity. Two out of three airplanes are operating using Thales’s equipment, whilst 40% of air traffic control centers around the world are supplied with Thales’s machinery. On top of that, the company produces sensors and mission systems, that are willingly exploited by navies and land forces. One of their most recent innovations is the “Defense and Innovation Frigate”, where the set of new sensors is integrated into the warship’s combat system and the recent version of Sea Fire radar would be installed to ameliorate the performance in the high-intensity battles. After a successful launch of their new product, the company is interested in the new research on improving marine communication between vessels or commercial ships.

MARITIMEMANET

MaritimeManet is the idea of the system that could create a Wi-Fi-based mobile ad-hoc network that supports broadband communication over large distances. The fundamental principle is to automatically discover neighboring nodes and to create a network with the strongest possible connection by choosing the best antennas on both ends. The system is using directional antennas for transmission, which is a major change from using omnidirectional antennas. The justification for the change is related to the need for a longer transmission range in the marine environment. Even though the initial idea is envisioned for the marine environment, it could be well used in places, where the speed of the information transmission is relatively slow or absent at all. So, besides the military, commercial use is also envisaged since it would also add benefits such as affordability and increased bandwidth. In the heart of MaritimeManet lays the algorithm, called Distributed Neighborhood Discovery (DND), which applies an analysis over received messages from the neighboring nodes to determine the strongest path to the node, that does not interfere with the rest of the network. The key is to do this procedure continuously since the ships or nodes are in constant movement.

WHY IS THERE A NEED FOR MARITIMEMANET?

There is no doubt that Thales Group already has technologies that support data exchange and communication in the marine environment. However, there is always room for improvement and their current machinery would require some modifications with the goal of making transmissions faster, more secure, and allowing the transfer of any type of data. The existing system that deals with the problem of communication between ships are Tactical Link System, however, it is not flawless. The disadvantages are the limitation to exchange of only predefined messages, low bandwidth, and the system, in general, is not cost-efficient. The desired switch is to find a solution, which would support the transmission of any type of data at high speed and stay affordable. In the Figure 1 down below, some alternatives and their drawbacks are represented in red color. The alternative in sense of satellite communication would be expensive and colossal delays in transmission are a major inconvenience that could cause troubles. Despite that, it would also mean the involvement of a third party, which is a gigantic disadvantage in case of a military use. Furthermore, this alternative is overly expensive for commercial use. Other alternatives are discussed in more details in the next chapter. So, the proposed idea promises to provide a reliable and fast network that would allow to transfer any type of data over large distances.

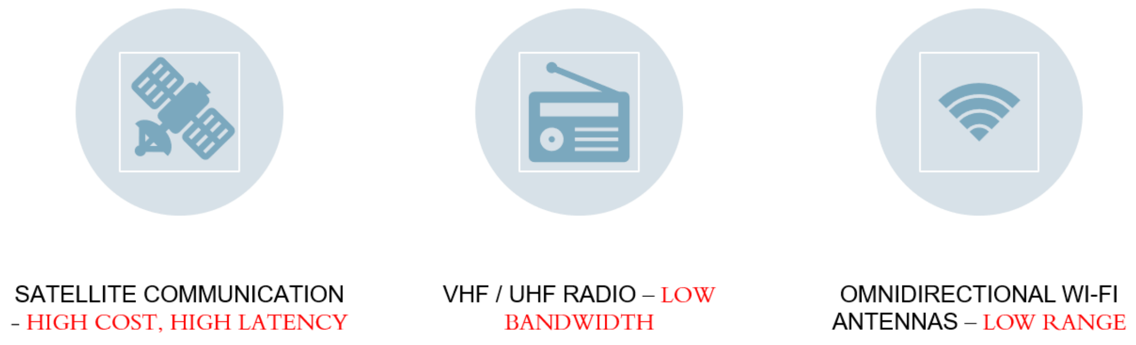

SATELLITE COMMUNICATION – HIGH COST, HIGH LATENCY
VHF / UHF RADIO – LOW BANDWIDTH
OMNIDIRECTIONAL WI-FI ANTENNAS – LOW RANGE

Figure 1: Alternatives and their drawbacks

EXPLORATION AND ANALYSES

In this chapter, the goals and expectations of the project are going to be identified and discussed. Also, the existing technology is going to be introduced, giving the brief overview of its concept. Since, this technology is not the only existing solution, it is going to be compared to alternatives to demonstrate its advantages to other options.

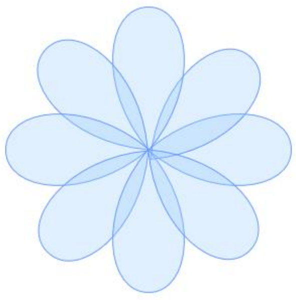
GOALS OF THE PROJECT

From the problems listed above, it can be seen that existing solutions are not adequate and require several modifications. The goal of MaritimeManet is to create a Wi-Fi-based mobile ad-hoc network that could support broadband communication. It involves the ability to support communication over large distances; detect new nodes; maintain and tracking of moving nodes; and monitoring of interference. To meet those expectation, there has to be a firm algorithm at heart of each node, that is a part of the network. The goal of the algorithm is to establish a strongest possible connection between each node on the network, in the form of a connected graph. Furthermore, it must come to the same conclusion of the strongest path on each end of the established connection. Additional goal would be elimination of connections between nodes that have a stronger 2-hop path.

The main problem to solve is deciding how the strongest possible wireless connection can be found and formed. The objective is to make sure that each node can consistently discover any neighbouring nodes, as well as decide which sector has the strongest connection to that node. Furthermore, maintaining the available set of frequencies is important, since the reduction of the interference is the key component to a solid connection between nodes. So, the goal of the project group is to implement algorithm mentioned above within the 10-weeks' time, based on the functional specifications provided by the client which determines and selects the most suitable antenna to establish a valid connection with a peer antenna.

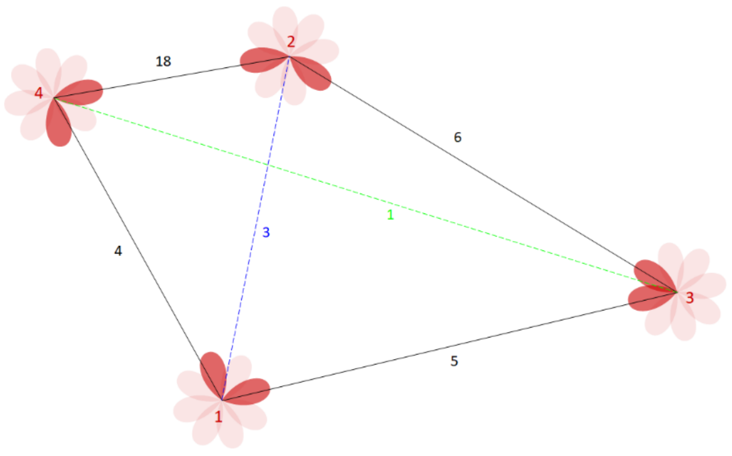
EXISTING TECHNOLOGY

The existing technology suggests using beam antennas also known as directional antennas to solve the problem of distance. The advantage of using a directional antenna is that it gives a longer spectrum for possible connection, hence a possibility for a stable connection with a large distance between nodes. Furthermore, it reduces the interference from other sources.



So, antennas are arranged in a sunflower pattern to cover 360 degrees. They are controlled from the hub, which is usually placed in the center of the sunflower within the radius of maximum 1 kilometer. The antenna is only active if there is a node that is in beam's radius. In the situation when the node is in the radius of multiple antennas, the algorithm that is running on the hub, must make a decision with an inclination towards the antenna with the highest signal strength. The algorithm is called distributed neighborhood discovery (DND), it makes decisions based on the "SENSE" messages, that each antenna periodically receives. Those messages carry a significant information from neighboring nodes, such as identification number of the node and its antenna, also called as sector, and strength of the signal. Each antenna has a neighborhood discovery (ND) component, that is responsible for retrieving information form "SENSE" messages and pass it to DND. After the DND makes decision, it send a signal to connection management (CM) part, also installed in the antenna, to establish a connection with a peer. The architecture is going to be explained in more detail in further Chapter 3.

To give a simple example of the existing solution, consider 4 nodes each consisting of multiple antennas and hub in the center. The red petals are active antennas, meaning involved in the wireless connections with other nodes. On the contrary, the transparent petals are not yet connected to any node. The number above the lines indicate the strength of the signal. Based on the "SENSE" messages, DND finds the candidates for connection and make a decision for each:



- Valid connection (solid black line) – DND will instruct to set up this connection using a frequency that results in minimum interference with other connections.
- Weaker connection (green dashed line) – Is a connection emanating from the same antenna a stronger connection exists. DND eliminates this candidate from the list.
- Slower connection (blue dashed line) – Is a connection between two nodes for which there is a two-hop path that is faster. DND eliminates this candidate from the list.

The result is a connected graph among the four nodes that optimizes the capacity and minimizes the number of wireless links needed for it. On top of this graph, or topology, a routing protocol is run to determine the forwarding tables at each node for proper multi-hop forwarding.

ALTERNATIVES AND COMPARISON

During the early years of the last century, radiotelegraphy using Morse code was the most reliable way to share the information at sea. However, in the age of automation, there were multiple drastic changes in marine communication to match the advancement of technologies. So, further down those alternatives are going to be shown and their drawbacks are going to be discussed. The information about each of the technologies was taken from Boat US Foundation.

CELL PHONE

Using cell phones in combination with VHF (very high frequency) proved to be one of the simplest ways to keep in touch with the land base. However, what is reliable on the land, may not be so reliable on the sea. Some characteristics of this alternative would not fit the idea of reliable long-distance communication. The range of cell phones stays in the line of sight, with the idea that originally cell phones are preferred to be used on the land. So, the ship has to remain relatively close to the shore if the cell phone is meant to operate as a way of communication. On top of that, most of the phones are still not water-resistant up to this day, which creates even more complications.

CITIZEN BAND RADIOS

These CB radios provide an economical alternative for two-way communication with 40 specific frequencies to the general public. However, this option is promised to be reliable only within 8 kilometres. So, vessels would have to be considerably close to each other to guarantee the reliable communication. Another pitfall of CB radios is the noise. Malfunctioning radio transmitters will lead to interference, which can create a dangerous consequence for the crew. Since the stakes are high, it is better to avoid unnecessary risk.

SATELLITE COMMUNICATION

Since the safety of crew, passengers, or cargo during the emergency on the sea, highly depends on the communication, the technology that provides it must be reliable in case of the any unforeseen danger. The satellite communication equipment operates outside of the mobile network, so in case of poor mobile signal, the important messages would still be able to reach the land base. Real-time data transfer is also a great advantage of satellites. Most of the modern equipment supports up to 124 kbits/s (Valčić, Mrak, & Gulić, 2016). However, the pitfalls of satellites are the high cost of services and lack of coverage in the areas of Earth's poles. On top of the that, the propagation delay may create the echo in case of telecommunication. The space events such as solar activity may affect the performance of the satellite and create an interference. Those drawbacks show that this alternative is not greater by any means than MaritimeManet.

CONCLUSION

Overall, with the fast growth of automation and fifth-generation of cellular network, telecommunication will adapt to the change. The alternatives mentioned above were great solution for its generation, however the introduction of Wi-Fi based mobile ad-hoc network is going to be a step towards the adaptation of new technological progress, which could help to eliminate propagation or transmission delays, to reduce interference and to secure reliability by creating a wider range for the signal .

PROPOSED ARCHITECTURE

This chapter presents the functional architecture of MaritimeManet. It does not describe the wireless propagation and transmission aspects. It solely describes the steps involved in making decisions between which antennas of which nodes a connection needs to be established. The chapter starts with the introduction of terminology that is vital for understanding the concept. Further, the description of the architecture will be presented from high level to more detail, mentioning each key component.

TERMINOLOGY

A **SENSE** message contains the following information:

- The identity of the own node. Generally, this is indicated by '*i*' but in reality, this identity must be unique within the network and can be derived from, e.g., the IP numbering plan.
- The identity of the own sector. Generally, this is indicated by '*k*' which denotes the position of the sector clockwise in the MBA.
- The current set of available frequencies {*f*} for sector *k*. The frequencies in this set are determined by the co-site interference with the existing connections at other sectors of the node. This means that the set of allowed frequencies is determined by the difference in frequency (spectral separation) and the difference in sector number (spatial separation) with other connections.
- The set of valid paths calculated in the previous period.

RP – radio paths – This repository contains radio paths as determined by and received from the ND component. A radio path is defined as a radio propagation channel between specific sectors *k* and *m* of two nodes *i* and *j*, respectively. The strength of this propagation channel is determined by (smoothed) received signal strength *s*. Radio paths form the basis for the calculation of valid paths. Radio paths have the perspective of the own node. Radio paths are updated every period. If there is no new update, then the age of the current sample is incremented. Radio path looks like (*i*, *k*, *j*, *m*, *s*). The following is interchangeably used in this document:

- *i* corresponds with own node identification 'onid'
- *k* corresponds with own sector identification 'osid'
- *j* corresponds with peer node identification 'pnid'
- *m* corresponds with peer sector identification 'psid'

VP – valid paths – Valid paths are candidates for connections. They are calculated by DND based on most recent radio paths and peer valid paths. The set of valid paths is a subset of the set of radio paths, more specifically it is the collectively determined subset of strongest possible propagation channels between sectors. Valid paths are the source for connection decisions made by the algorithm and are updated every period. If there is no new update, then the age of the current sample is incremented. Valid paths have the same perspective as radio paths.

pVP – peer valid paths – This repository contains the valid paths as calculated by and received from peer nodes. The peer valid paths are needed for the calculation of the own node valid paths. Notice that, when carried in SENSE messages and received by nodes, the perspective of peer valid paths is that of the sending peer node! Peer valid paths are updated every period. If there is no new update of a peer valid path, then the age of the current sample is incremented. If a complete set is missing, i.e., a corrupted SENSE message, then the age of the complete set of current peer valid paths is incremented.

C – connections – This database contains the current connections of the node with peer nodes. The state of the connections, i.e., what is the strength of the connection and which sector is involved, is updated every period. If there is no update for a connection, the age of this connection is incremented.

Hub – is a powerful single-board computer, that represents the center of the MBA. The DND algorithm is running on the hub.

MBA – Multi-Beam Antenna is a combination of multiple antennas, each representing a sector of the node.

Control radio – a channel used for receiving and sending SENSE messages only.

Data radio – a channel used for the exchange of user data.

DND – Distributed Neighborhood Discovery – algorithm that takes care of selecting the best possible antenna for a connection. It makes a decision based on the information received from the SENSE messages. The algorithm includes Path Management and Link management, that are going to be described further in the document.

FROM PHYSICAL TO FUNCTIONAL ARCHITECTURE

In its simplest form, the physical architecture of a MaritimeManet node is depicted in Figure 2. The left part shows the spatial arrangement of the sectors. Each sector contains a single board computer (SBC) with OpenWRT operating system. The SBC has an Ethernet interface to connect it to the switch and can accommodate two wireless 802.11 interfaces: one is used for sending and receiving SENSE messages (control radio), the other is used for exchange of user/application data (data radio). Each 802.11 interface is connected to a directional antenna radiating outwards such that all antennas collectively cover the complete azimuth. The passive switch is used to multiplex the connections from the sectors and connect them to the HUB. The HUB is a powerful SBC also running OpenWRT. It contains the following capabilities:

- the DND application which controls the wireless connections among sectors.
- a BATMAN process that takes care of routing frames.
- an Open vSwitch process that is part of connection management.

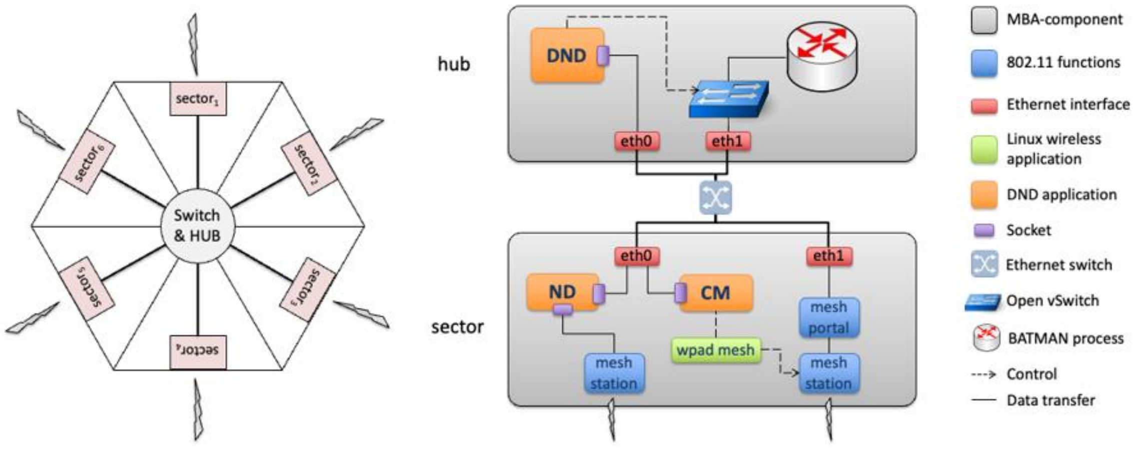


Figure 2: Physical arrangement (left) and decomposition (right) of a MaritimeManet node

The mapping of MaritimeManet's key physical aspects to control plane functionality is shown in Figure 3. It shows the decomposition of one of the sectors of the Multi-Beam Antenna (MBA): the Neighbourhood Discovery (ND) process that interfaces with the control radio and the Connection Management (CM) process that configures the data radio. The control radio of all sectors in a MaritimeManet operate on the same pre-defined control frequency. This is needed to secure that control information sent by each sector can be received by all sectors within radio coverage. In fact, the control radios of all sectors of all nodes of the network are configured with the same mesh basic service set (MBSS). The data radio of each sector is used to set up a data connection with a sector of a peer node. All data connections collectively form the network over which user and application data is exchanged. The radios all reside in the MBA, which is deployed at suitable, i.e., high enough, positions of the carrying platform. The control information processing and routing of data is performed at the HUB, i.e., a sufficiently powerful computer if needed in a conditioned location at the platform. The HUB and the MBA collectively form the node.

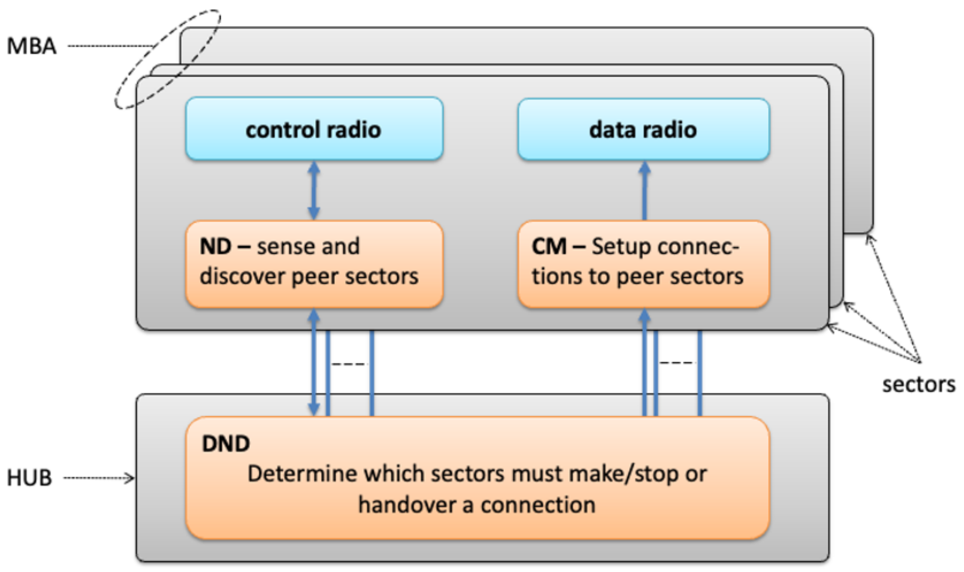

MBA
control radio
data radio
ND – sense and discover peer sectors
CM – Setup connections to peer sectors
sectors
HUB
DND
Determine which sectors must make/stop or handover a connection

Figure 3: Mapping of the physical deployment to the high-level functional architecture.

NEIGHBORHOOD DISCOVERY

The Neighbourhood Discovery (ND) component consists of two processes that run at each sector. The first process periodically composes SENSE messages and broadcasts them via the control radio. The period ranges from 100ms to 500ms. The SENSE message is the vehicle that enables discovery of new nodes in the neighbourhood and tracks the presence of discovered nodes.

The second process receives SENSE messages broadcasted by peer nodes via the control radio. For this purpose, control radios continuously listen for the SENSE messages from peer nodes. Upon receiving a SENSE message from peer sector *m* of peer node *j*, the ND component retrieves the strength *s* of the received message from the radio driver. Per own sector and per peer sector, ND maintains a filter, e.g., an alfa filter, to smoothen the variations in the received strength. The smoothened strengths are then stored together with the node and sector IDs as a radio path.Although SENSE messages are received from peers, radio paths obtained from these messages are viewed from the perspective of own node in the sense that own node IDs are mentioned first in the notation.

CONNECTION MANAGEMENT

The radio paths determined at each sector of a node are handed over to the Distributed Neighbourhood Discovery (DND) component for further processing. DND processing eventually results in a decision whether or not sectors of two nodes must be connected and at which frequency. Since DND is a distributed algorithm, the peer node arrives at the same conclusion and the connection can be established without message exchange. This decision is handed over to the Connection Management (CM) component of both nodes.

Execution of connection management is partly done at the HUB and partly at the sector. The former configures the Open vSwitch such that the VLAN of the sector that must be connected is switched to the BATMAN interface. The latter is done by the CM component at the sector. It configures the data radio within the MBA in order to actually establish the connection. This connection terminates at only one sector of precisely two nodes, i.e., it is ‘dedicated’ to only these sectors. This is a powerful restriction of MaritimeManet since it avoids the throughput degrading ‘hidden sectors’ operating the CSMA/CA protocol of 802.11 networks! Especially in networks with directional antenna’s, such as MaritimeManet, having more than two sectors in a connection lead to the ‘hidden sector’ problem, which is similar to the well-known ‘hidden node’ problem.

FUNCTIONAL DECOMPOSITION OF DND COMPONENT

The first-level decomposition of the DND component is shown in the grey rectangle in Figure 4. In green are shown the various processes that make up the DND and that run in the HUB of a node. In blue the needed data repositories or databases are shown, which contain the relevant data records for DND.

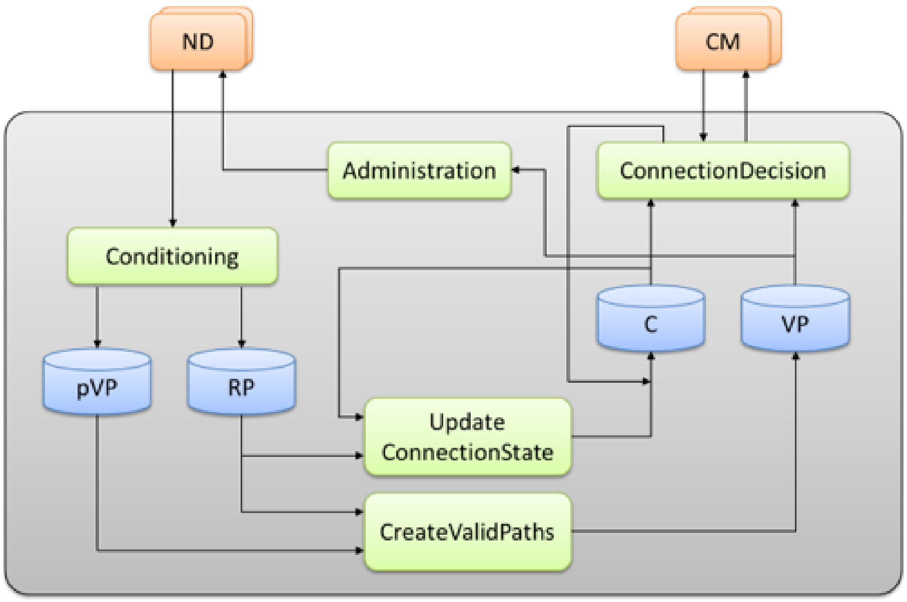


Figure 4: Decomposition of the central DND component

CONDITIONING

This component takes the determined radio paths (*i*, *k*, *j*, *m*, *s*) and peer valid paths carried in the SENSE message from the ND component. The radio paths are stored in database RP according to well-defined data structures. In addition, the peer valid paths are stored in the database pVP. Before storing these peer valid paths, duplicates are removed. Duplicates will occur because SENSE messages are received at adjacent sectors of the node, and although this leads to different radio paths, the carried peer valid paths are the same.

UPDATECONNECTIONSTATE

This component reads the most recent radio paths from RP, and the current connections from C. For each connection whose (onid, osid, pnid, psid) matches with a radio path, the strength will be updated, and the age will be reset. For each connection without such a matching radio path, its age is incremented. The connection is deleted when the age is larger than a pre-set value. Logical explanations for a connection not having a matching radio path are:

- the distance has increased beyond which a connection can be supported.
- an object has obscured the line of sight between the nodes.
- the peer node has a failure.

ADMINISTRATION

This component determines, for each sector of the node, the currently available frequencies that must be conveyed in the next SENSE message for that sector. The current valid paths are added to the next SENSE message of all sectors.

CREATEVALIDPATHS MODULE DECOMPOSITION

The component CreateValidPaths is internally structured according to Figure 5. This component is the heart of the DND process. It turns observed radio paths and received peer valid paths into a set of valid paths, each of which is a candidate for a wireless connection between the nodes. Largely the valid paths are created in two steps: first the 1-hop valid paths are determined (Path Management), and second the 2-hop optimizations are performed (Link Management). The role of the peer valid paths is essential in both.

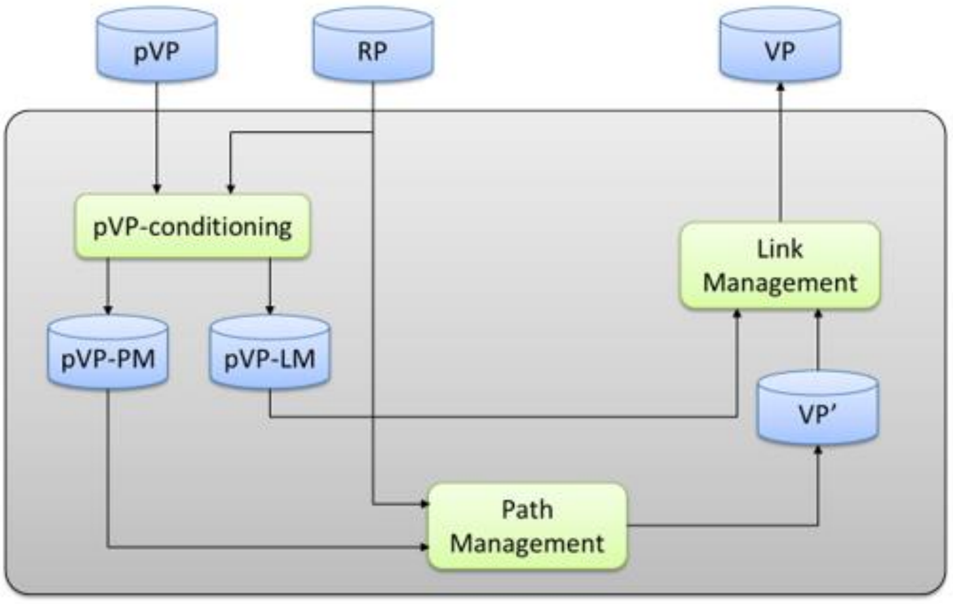


Figure 5: Decomposition of the CreateValidPaths component

PVP-CONDITIONING

The peer valid paths must be split in two disjoint subsets. Only those peer valid paths are selected for Path Management whose originating sectors are equal to a peer sector in the radio paths. Before storing them internally, these peer valid paths are made dual. This means that the originating and terminating sectors are swapped: $(i,k) \leftrightarrow (j,m)$. Now they are easy comparable with radio paths, i.e., have the same perspective as the radio paths of that node. These dual peer valid paths are stored in pVP-PM.

The remaining peer valid paths form the subset that is needed for link management. Hence the name of the data structure pVP-LM they are stored in. These peer valid paths are characterized in that they have no own sector in common with the peer sector of a radio path.

PATH MANAGEMENT

In path management the dual peer valid paths from pVP-PM and the radio paths are merged in one list that is ordered by strength. The role of the dual peer valid paths is to check if their weight is higher than a radio path terminating at the same peer sector. In that case this radio path cannot be a valid path.

Path management now performs the following selection of the valid paths. The strongest path, i.e., the first path, can only be a valid path if the 'onid' equals the ID of the node (this seems superfluous but for dual peer valid paths this need not be true). If so, the corresponding 'osid' and 'pnid' are added to the used-osid and used-pnid sets. For all subsequent paths, the following is checked:

- Is the 'onid' equal to the ID of the node?
- Is the 'osid' not in the used-osid list?
- Is the 'pnid' not in the used-pnid list?

If all checks are affirmative, this path can be added to the list of temporary valid paths, i.e., to the VP' database.

LINK MANAGEMENT

In principle the valid paths that have been found can be input to the 'ConnectionDecision' component. However, an optimization is possible which is advantageous. Suppose there exists a 2-hop path to a peer node that has more capacity that a single hop path to that same peer node. In that case the direct connection is not needed and thus its corresponding wireless transmission resources can be devoted to other connections that need it more. 'Link Management' achieves this optimization by comparing the strength of each temporary valid path with the composed strength of a 2-hop path to that same node, if such 2-hop path exists. If this 2-hop strength is larger, the direct link is not needed, and this temporary valid path is not stored in VP.

CONNECTIONDECISION MODULE DECOMPOSITION

The decomposition of 'ConnectionDecision' component is shown in Figure 6.The main tasks for this component are to periodically update the connection database and to define the corresponding decisions regarding the connections on the data radios.

COMPARE & DECIDE

In this module each valid path is compared to the current connections. For each valid path it decides what to do: make a new connection, continue an existing connection, hand over an existing connection, or delete a connection.

When there is no matching connection for a valid path, a new connection must be set up between the sectors of the valid path. The frequency for this connection is unambiguously determined from the non-empty cross-section of the available frequency sets of the concerned sectors. If this cross-section is empty, the connection cannot be established[1].

When the peer node of an existing connection matches with the peer node of a valid path, the connection can be continued as is. If, in addition, the own sector of the valid path differs the own sector of the connection and is stronger than the current connection by a pre-set value, this connection is handed over to the own sector of the valid path. The frequency of the handed over connection remains the same, and this frequency needs to be available at the new sector.

All decisions are handed over to the CM component where they are effectuated.

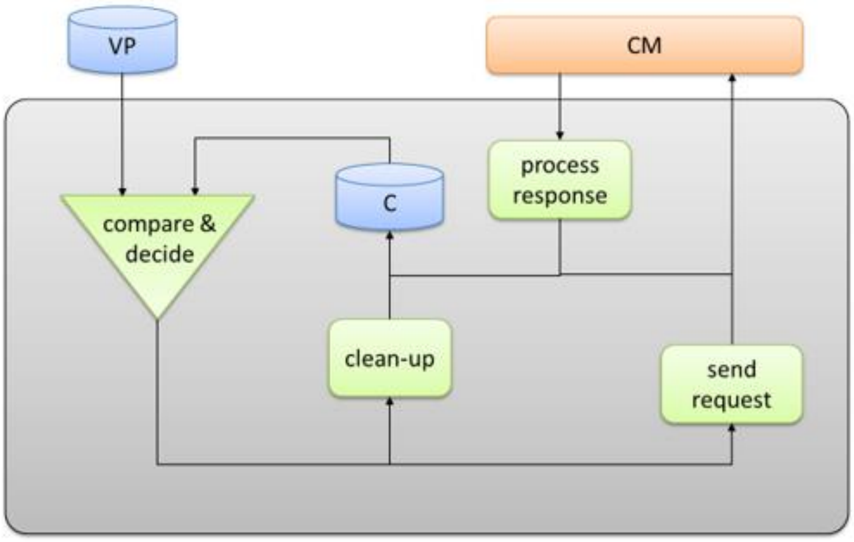


Figure 6: Decomposition of the ConnectionDecision component.

CLEAN-UP

All connections that have not been continued or handed over by the compare & decide module are eligible for disconnection. More specifically, when their age is larger than a pre-set value, the connection will be disconnected. These were connections with nodes that have lost radio coverage for reasons mentioned in "UpdateConnectionState" or have changed to another node with a stronger connection.

[1] It is possible to check the non-emptiness of this cross-section upfront and remove the concerned radio paths.

REQUEST AND RESPONSE

The decision made in module compare & decide leads to a command for the corresponding sector containing the connection decision made. CM will effectuate this decision and the outcome of that effort is reported back to ‘ConnectionDecision’

SCOPE AND REQUIREMENTS SPECIFICATIONS

This chapter will introduce necessary requirements to meet the expectations of the client for a working product. In the beginning, the requirements were gathered from the information about the functional architecture of the system. Further, more details were found through weekly progress meeting with a client. Moreover, some of them were altered or removed due to the change of scope and client's feedback on the progress. The chapter will go deeper into the details of what the scope was during the starting phase and why it changed during the development. It will also mention how the change affected the requirements.

INITIAL SCOPE OF THE PROJECT

The initial scope of the project was to make use of the finished ND and CM algorithms and implement the DND algorithm for the real hardware.The Neighborhood Discovery (ND) and Connection Management (CM) algorithms will be running on the sector while the DND algorithm will be running on the hub. The ND algorithm is responsible for sending and receiving SENSE messages to discover peer sectors through the control radio. SENSE messages are like HELLO messages, to notify the peers that they are reachable. The data output from the ND component is then used by the DND algorithm to determine which sectors must make/stop or handover a connection. The decisions taken by the DND algorithm will then be used by the CM component to establish real data connections between the nodes through the data radio. It also must manage the frequency channels, so that each sector is using a unique channel for its connection with its peer node sector. So, nodes must exchange their available channels and choose a common free channel on both sides.

SYSTEM REQUIREMENTS

The functional requirements of each component are specified as follows:

ND:

1. *The ND (Neighborhood Discovery) must send/receive SENSE messages.*
2. *The ND must determine radio paths from received SENSE messages.*
3. *The ND must supply radio paths to DND (Distributed Neighborhood Discovery).*
4. *The control radio must use the same frequency among all sectors of all nodes.*
5. *The control radio must continuously listen for SENSE messages on the same frequency.*
6. *A SENSE message must hold the id of the node, id of the sector, set of available frequencies, and set of valid paths.*

DND:

1. *For **path** management, the DND must select only those peer valid paths, whose originating sectors are radio path peers.*
2. *For **link** management, the DND must select only those peer valid paths, whose originating sectors are **not** radio path peers.*

3. *DND must compare a new valid path with the set of existing connections and decide what to do:*
4. *If a valid path does not match any existing connections, make a new connection,*
5. *If a valid path matches an existing connection, continue an existing connection,*
6. *If nodes of valid path and existing connection match, but sectors do not, then hand over an existing connection,*
7. *If no actions above are taken, then drop the connection.*
8. *The connection cannot be set up if the list of frequencies is empty.*
9. *All DND decisions are handed over to CM (Connection Management) component.*
10. *All connections that have not been sent to CM are eligible for disconnection, and the outcome of this action must be reported back to DND.*
11. *At every period:*
    a. *radio paths,*
    b. *valid paths,*
    c. *Peer valid paths are updated.*
12. *If a connection does not receive an update, its age is incremented.*
13. *If a connection's age is larger than a pre-set value, it is dropped.*
14. *If there is a stronger 2-hop connection, the system must discard the direct link, and connect via 2 hops instead.*
15. *The DND must send a set of available frequencies to ND*
16. *The DND update frequency must be universally controlled.*
17. *The DND must send current valid paths to ND*

CM:

1. *Each sector of each node must have at most one active connection.*
2. *CM must be able to make a new connection.*
3. *CM must be able to change a connection.*

CHANGE OF THE SCOPE

The type of functionality to be implemented required a lot of testing and using the actual hardware for that was going to add an extra layer of complexity to the project and may deviate the team from finishing the main goal due to hardware related issues. As a result, the project's scope has been changed from working with the hardware to implementing a simulator, a simulator handler, and a drawing. This has resulted in easier visualization of the network topology and composing a simulation video to carefully observe the network behavior and be able to assess whether it is behaving correctly or not.

The simulation has replaced the ND component so instead of having an ND component on each node's sector, nodes can know how the world looks like through the simulation. The simulator generates the state of the world (the radio paths) for each timestamp and the simulator handler retrieves the radio paths for that time stamp and then distributes them to the clients/nodes. Then each of the clients runs the DND algorithm on its side resulting in a list of valid paths. The clients send this list to the simulator handler then the simulator handler sends the new radio paths and uses those lists of received valid paths to distribute them again as peer valid paths, so that each node knows what the valid paths for its peers are, which are needed for the DND algorithm. In order to visualize the network, initially a visualizer component was used to draw only the radio paths for easier illustration of the reachable nodes. Furthermore, a drawing component was implemented to receive the valid paths for each time stamp and converts this into a simple drawing as shown in Figure 7.

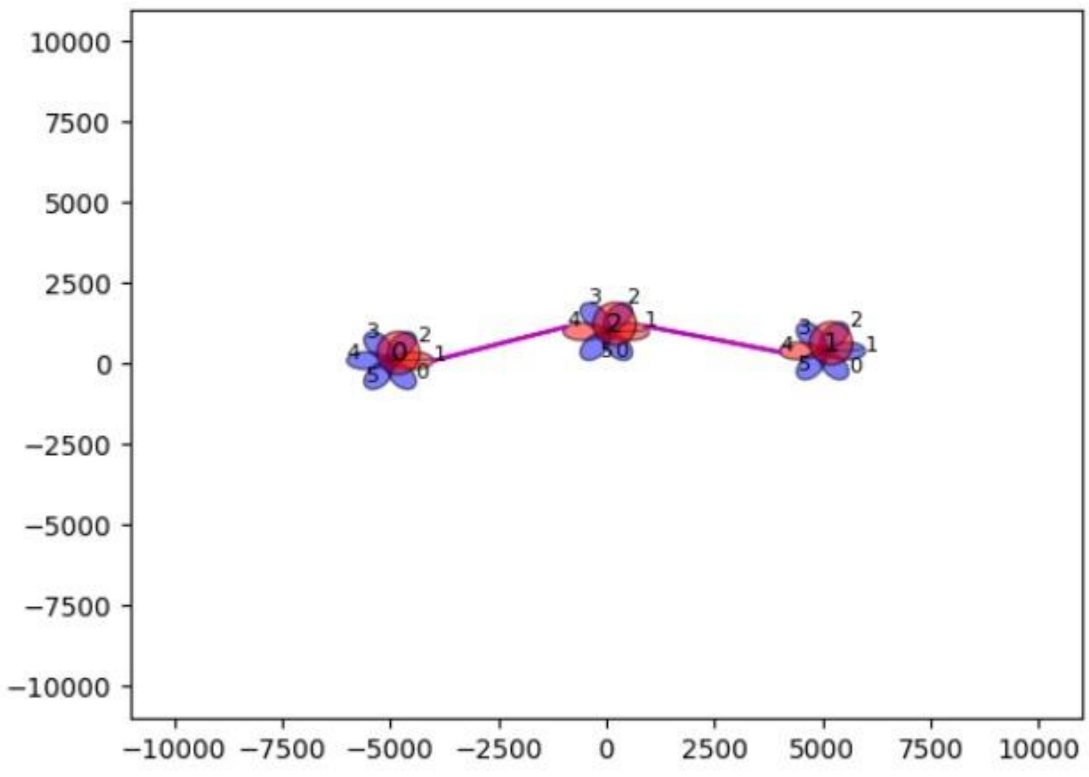


Figure 7: Example of drawing

NEW REQUIREMENTS FOR THE CHANGED SCOPE

The functional requirements for the simulation are specified as follows:

SIMULATOR:

1. The simulator must be able to generate radio paths for a given number of nodes and sectors
2. The simulator must be able to generate a number of states of the world according to a given number of iterations

SIMULATOR HANDLER:

1. The handler must be able to connect with a given number of clients/nodes
2. The handler must be able to retrieve the radio paths for each iteration from the simulator
3. The handler must send the radio paths correctly to each of the clients/nodes
4. The handler must be able to receive the nodes’ valid paths

5. The handler must be able to send back the radio paths and peer valid paths correctly back to the nodes
6. The handler must send the valid paths to the visualizer for drawing
7. The handler should be able to check if the network has converged

VISUALIZER/DRAWING:

1. The drawing must be able to draw the correct number of nodes and sectors
2. The drawing must be able to place the nodes in the correct coordinates
3. The drawing must be able to draw the correct valid paths between nodes
4. The drawing must be able to update the node and sector position
5. The visualizer should be able to form a video of the whole simulation

SIMULATION FLOW

Firstly, the figure below represents the flow of the simulation. Those components are going to be described in detail in the next Chapter 5. However, the key idea to grasp is that the *Sim_handler* component takes a state of the world for each timestamp from the *Simulator (mama_sim*) and sends only those RPs and/or PVPs that are relevant to that client. So, clients cannot see what connections are available for the node that is not their neighbors. Furthermore, each client runs the DND algorithm to make a decision for the connection. To visualize each state of the world and make sure that the decision of connected nodes was mutual, there is a *Drawing* component, that makes a graph at each timestamp and updates if the topology has changed.

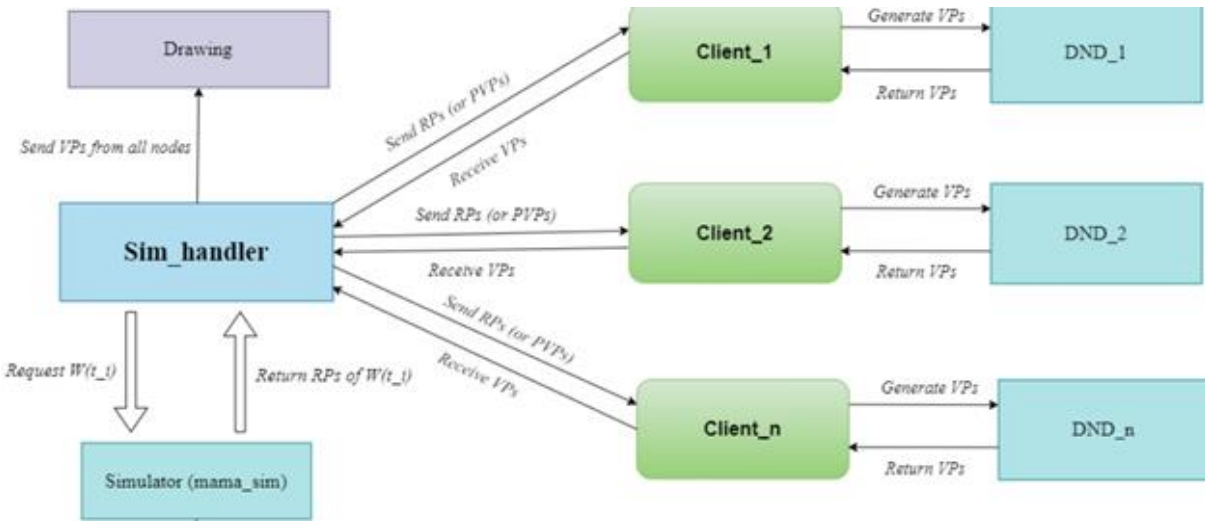


Figure 8: Simulation flow

FROM SIMULATOR TO REAL SYSTEM

Figure 9 provides a clear picture of how the simulator is related to the real-world system. From the figure, it can be seen that the *Sim-handler* component represent ND, since it holds every RP or PVP (basically the state of the world during timestamp *i*). As it was stated in the requirement that the DND has to run on the hub, simulator acts as each client is a hub, since it is also running the DND. This simple graph gives a basic idea behind how the simulator could be future translated into the system that can be applied in the real world.

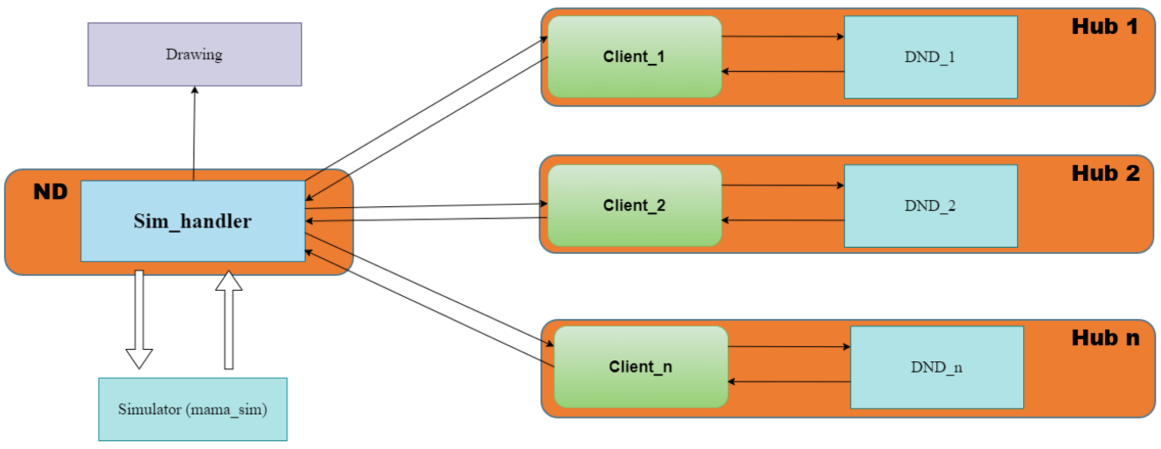


Figure 9: Mapping of the simulator to the real-world system

DESIGN OF COMPONENTS

Current chapter describes the design phase and decisions that have been made to achieve a working product. It will start with the explanation of why the current infostructure is desired and what choices were made to have a solid base for a start. Further , it will discuss the design of each component, highlighting the reasons for the choice of implementation. In the end, it will provide some additional information, showing more technical details, that are optional for understanding of the reader.

PRELIMINARY DESIGN CHOICES

To have a solid base for the project, it is important to make the right decisions from the beginning. It includes choice of the programming language, frameworks and libraries, communication between the multiple processes and the execution. Further down, the most important decision are going to be elaborated.

PROGRAMMING LANGUAGE

Python has been chosen as the programming language for this project. *Python* is an easy to learn but powerful language that suits very well with the nature of the project. The previous version of the system is written in *C*, which suits embedded computers well, because of how efficient it can be with memory. However, for simulation drawing purposes *C* is verbose and not universally used. On the other hand, *Python* has an exceptionally large community of engineers who use it for visualization which means more available information online. Moreover, *Python* is a very concise language that can drastically improve development pace, which is an important consideration due to how time-limited the project is.

FRAMEWORKS AND LIBRARIES

The team settled for *Matplotlib* as the drawing library. This library is usually used for graph drawing in Python. It covers all requirements regarding visualization – ability to draw nodes with their identifier, draw sectors with their number, highlight sectors with active connections, draw paths between sectors, etc.

For mathematical computations, the chosen library is *NumPy*. This is used to do calculations in the simulation – calculate antenna losses, node locations, etc. *NumPy* is an industry standard library providing high-level mathematical functions to operate on numbers and matrices.

INTER-PROCESS COMMUNICATION

The system consists of various components that run in parallel and communicate. The means of process communication is UDP sockets. Sockets are a simple yet powerful way to send data between processes or computers. When communicating with sockets, the processes send messages that need to be handled in software. It is the programmer's responsibility to define a protocol of how the messages are sent. There must be some way to identify what type of message is sent, what the data is, etc.

EXECUTION PARADIGM

The main parts of the system – the sim handler and the clients follow a periodic execution paradigm. At each period, each sub-system executes the same set of actions in sequence. The initial actions are receiving of data, following the handling of the data, and lastly sending the results.

SOFTWARE COMPONENTS

To avoid bewilderment from the complexity of each component, a great design is a key for a simplification and clear understanding for those who is not familiar with the concept of the project. There are 4 important components that create the entire system. Each of them is going to be described separately, elaborating on decisions of the design.

SIMULATOR

The core of the built system is the simulator. The simulator is responsible for the locations of the nodes and the calculation of radio paths between all pairs of nodes and their sectors. The team was provided with a reference implementation in *MATLAB* and the goal was to translate this code to *Python*. There are multiple ways implemented, how the sim handler can interact with this component.

- Option 1: the simulation is run for a predefined number of iterations "*N*". First, the simulator generates the locations and orientation of the nodes, as well as their velocity, direction, and angular velocity (velocity of rotation). Then, the simulation computes the parameters for all nodes for all *N* time steps. This was useful to test whether the *Python* implementation matches the *MATLAB* reference. Only used in the initial stages of the project, this is deprecated and not used for the simulation of the final system.
- Option 2: the simulation is called with given node parameters to calculate the node parameters for the next time step. This can be used both for a random scenario and a predefined scenario of the world.

SIM HANDLER

The sim handler component is the coordination center of the system. It is responsible for requesting the simulator to set up the world (or loading a predefined scenario), then waits for the required number of clients to connect. Once all clients are connected, at each time step the sim handler executes the following actions:

1. Handles the incoming messages from clients – extracts the valid paths that the clients formed in the previous time step,
2. Sends these valid paths to the drawing process,
3. From the simulation, requests the node radio paths for the next time step
4. Sends the radio paths and peer valid paths to each client.

Note that each client only receives the radio paths and peer valid paths they would like with the real ND (Neighborhood Discovery) component. This way, there is no fully observable state of the world for any of the clients.

CLIENT

The client component represents a hub in the real system. The hub receives radio paths from each of the sectors. The simulation replicates this behavior.
When started, the client first connects to the sim handler, the sim handler replies with their unique client identifier. Once the client has received their id, it executes the following actions periodically:

1. Handles an incoming message from the sim handler – extracts the radio paths and the peer valid paths,
2. With the received radio paths and peer valid paths, runs the DND (Distributed Neighborhood Discovery) algorithm (PVP conditioning, path management and link management),
3. Sends the resulting valid paths back to sim handler.

DRAWING

The drawing component is used to visualize the nodes and the connections. At each period, the sim handler sends the valid paths of all nodes to the drawing component. These valid paths are then filtered in the following way, for each valid path, we check whether the peer node of that valid path chose the same path to connect to that node – if it did, then that is a two-way connection – meaning in the real system, these nodes would establish a connection.
The nodes are then drawn as well as the connections. The sectors used for the connections are highlighted in red, the inactive sectors are drawn in blue.

ADDITIONAL DESIGN INFORMATION

This section is design for a more technical elaboration, that could interest the reader. It will provide more detailed insights into the system.

INTER-PROCESS COMMUNICATION – DETAILS

The components use UDP sockets for communication. The main socket used is at the ***Sim Handler***. When started, the ***Sim Handler*** opens a socket on the local machine, with port **5005**. It then listens for incoming messages and expects each client that is started to send a **HELLO** message. Once the handler receives a **HELLO** message, it replies with the node's unique identifier. If the simulation is configured for 4 nodes, then 4 clients must connect and they will receive identifiers 0, 1, 2 and 3 in the order of connection.
When the client is started, it opens a socket on a randomly selected port, and sends a **HELLO** message to the handler. It is expected that the sim handler is already running and is waiting for such a message from the client.
Lastly, when the drawing is started, the component opens a socket on port **5000**. At each time step the sim handler sends the valid paths of each node to the drawing. The drawing component then extracts the valid paths from the message and draws the resulting connections.

DESIGN OF INTEDED USE

First, the user must enter the starting parameters in the configuration file. The configuration file is a *JSON* formatted file, that receives the parameters of the simulation such as:

- **N** – number of nodes,
- **M** – number of sectors per node,
- **F** – update frequency of Sim Handler and Client in Hz (executions per second),
- **O** – the X and Y position for each node as well as the rotation angle,
- **vtran** – the velocity of each node in units per time step,
- **otran** – the direction of velocity vector with respect to the zero angle,
- **vrot** – the angular velocity of each node in radians per second.

An example configuration file looks like this:

```
{
"N":3,
"M":6,
"F" :4,
"O" :[[-5000,100,0],
        [5000,400,1],
        [0,1000,2]],
"vtran" :[[10,20,30]],
"otran" :[[0,2,1]],
"vrot" :[0,0.1,0.05]]
}
```

In this case the simulation will be started with 3 nodes, 6 sectors per node, update frequency of 4 Hz, node locations:

```
    Node 0: x=-5000, y=100, angle=0 rad
    Node 1: x=5000, y=400, angle=1 rad
    Node 2: x=0, y=1000, angle=2 rad
```

Translation velocity:

```
    Node 0: 10 units/time step
    Node 1: 20 units/time step
    Node 2: 30 units/time step
```

Translation direction (w.r.t. zero angle):

```
    Node 0: 0 rad
    Node 1: 2 rad
    Node 2: 1 rad
```

Angular velocity:

```
    Node 0: 0 rad/time step
    Node 1: 0.1 rad/time step
    Node 2: 0.05 rad/time step
```

Optionally, the user can only specify the number of nodes, sectors per node and the update frequency to run a random scenario. The configuration file will then look like this:

```
{
"N":3,
"M":6,
"F" :4,
}
```

TESTING

After the requirements were introduced and approved by the client, they must be met and integrated into the project. To verify that the integration behaves as expected, it must be tested in various ways that could prove its accuracy. This chapter will introduce the chosen approach for testing that ensures that the requirement is indeed present in the system. Furthermore, to guarantee that the simulation performs correct calculations and does not make unexpected results, the testing of the functionality of the system will also be explained. Lastly, client satisfaction is an important detail, that must constantly be confirmed. The chapter will show how the system has changed after the client's feedback.

TESTING APPROACH

For the verification of the requirements, four fundamental methods were considered to decide which way of testing would be the most suitable for each requirement. Those methods are **I**nspection, **D**emonstration, **T**est and **A**nalysis. Down below, a description for each type is presented:

Inspection - it would include the use of five senses (visual, auditory, olfactory, tactile, taste) to assess if the product contains all the physical characteristics it is supposed to have as mentioned in the requirement.

Demonstration – it would include the verification that the product behaves as it was asked to. Meaning that all the functional requirements introduced by the client demonstrate expected results as it was mentioned in the requirement.

Test - it is a more precise and calculated form of the demonstration. It would require a predefined data as an input to perform some calculation or apply an algorithm to show a very specific expected outcome, which could be compared to a number or a predefined scenario.

Analysis - it is a process of applying calculations and models to validate a product or system. Based on some representative, actual test findings, analysis is used to create predictions about a product's or system's performance. Without resorting to destructive testing, it can be used to calculate failure points based on actual test findings.

DESCRIPTION OF TESTS

Since the scope of the project was changed to the implementation of the simulator, only requirements for the simulator were considered for the testing purposes. Each of the requirement, that is separated to the component's group, is given a type of testing and its detail description of how the test should be performed and which results are expected upon the end of the test. Everything is organized into the tables per each component for a convenience of the reader.

SIMULATOR

The requirements for the simulator component and their tests are presented in the table below:

Requirement	***Method of Verification***	***Description***
The simulator must be able to generate radio paths for a given number of nodes and sectors.	D	To verify that simulator generates radio paths, solely demonstration would be suitable. Since radio paths are calculated from the pre-given matrix that represents state of the world, a final list of RPs should be filled with RPs for each node and each timestamp.
The simulator must be able to generate a number of states of the world according to a given number of iterations	T	A test could calculate how many lists of RPs (as per each timestamp) are in the final list and the number must be equal to the number of iterations.

SIMULATOR HANDLER

The requirements for the simulator handler component and their tests are presented in the table below:

Requirement	***Method of Verification***	***Description***
The handler must be able to connect with a given number of clients/nodes	A	Firstly, a test must be set up to check if the expected number of connection equals to the number of clients with the simulator. Secondly, there must be an analysis that check if the clients connected successfully, such as "Hello" message was delivered.
The handler must be able to retrieve the radio paths for each iteration from the simulator	T	A test could be set up, that checks if the retrieved RPs from each iteration are the same as they are expected to be.
The handler must send the radio paths correctly to each of the clients/nodes	A	"Correctly" in this case means – send only those RPs to the client that belongs to it (so origin of RP must be client's ID). Each client must have a test that check if the RPs it

		received are equal to the RPs it expects. After, the analysis should be made to check if all the clients' tests are passed
The handler must be able to receive the nodes' valid paths	D	A demonstration would enough to prove that the VPs are received by the handler. A check could be made that shows that VPs sent by the client are the same are received by the handler.
The handler must be able to send back the radio paths and peer valid paths correctly back to the nodes	A	"Correctly" in this case means – send only those RPs and VPs to the client that belongs to it (so origin of RP or VP must be client's ID). Each client must have a test that check if the RPs and VPs it received are equal to the RPs and VPs it expects. After, the analysis should be made to check if all the clients' tests are passed.
The handler must send the valid paths to the visualizer for drawing	T	A test could be set up, that checks if the received VPs are the same as they are expected to be.
The handler should be able to check if the network has converged	T	A network has converged if each path such as *(i,k,j,m,s)* has a pair such as *(j,m,i,k,s).* Basically, if the path start at node ***k*** with sector ***i*** and ends at node ***m*** and sector ***j***, then there should be a path that starts at node ***m*** and sector ***j*** to the node ***k*** and sector ***i***. To check if the network converged, there should be a test that check if each path has its pair.

VISUALIZER/DRAWING

The requirements for the visualizer/drawing component and their tests are presented in the table below:

Requirements	***Method of Verification***	***Description***
The drawing must be able to draw the correct number of nodes and sectors	I	The best way is to visualize the nodes that are present on the screen and check if they are

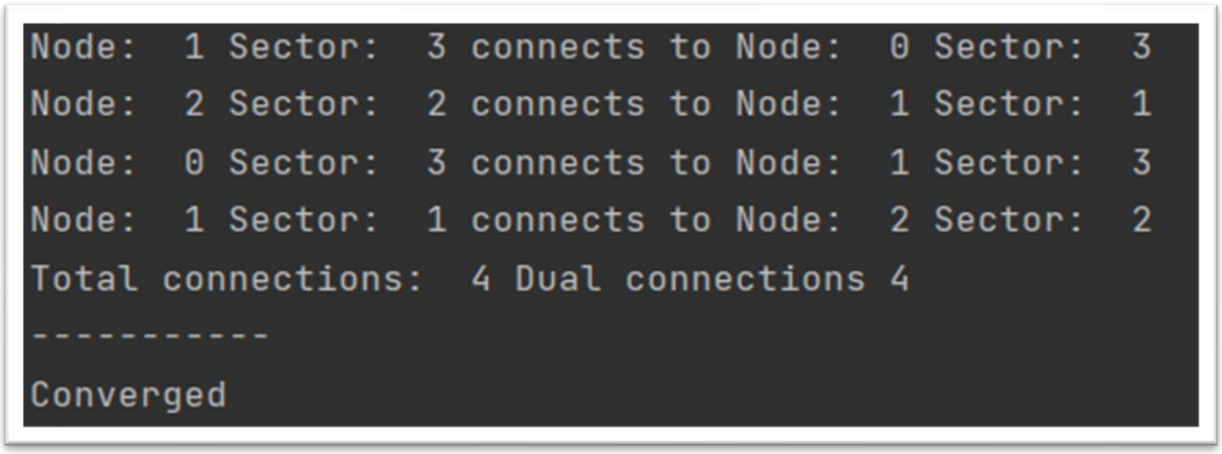
		equal to the number of nodes participating in the simulation.
The drawing must be able to place the nodes in the correct coordinates	T	A test can be set up that check if the coordinates of the nodes are equal to the ones, that were given to the simulator.
The drawing must be able to draw the correct valid paths between nodes	I	The best way is to show that VPs that are resulted from the clients' DND are present on the screen, also check if there are no extra lines between nodes.
The drawing must be able to update the node and sector position	A	There should be analysis that check if the coordinate of node is changed so should be the graph. Also, a visual check that the drawing is not static and the movement is present.
The visualizer should be able to form a video of the whole simulation	I	The basic way is to check if the video file is present and not corrupted. Also, to check if the all the nodes and their connections are constantly present in the video.

EXAMPLE OF THE TESTING FUNCTION

A testing function was implemented to test the convergence of the network on the simulation handler side (ND), in other words, the goal for the function was to compare the number of dual connections against the number of total connections. The function also outputs the number of dual connections, so it was beneficial in the beginning few weeks when visualizing the network was not possible. Having the testing function alongside the visualizer have made testing more compact and eliminated the bugs that might emerge from the visualizer.

An example of how the function works is illustrated on the Figure 10. This picture shows the connections received by the simulation handler from the clients and there are 4 connections which are identical/dual. This Figure 11 shows the resulting topology of the testing scenario, however there can be seen only two (basically 4, since each connection has a pair) connections, which can be explained that the node 0 and node 2 did not come to agreement to which sector of each other they must connect.

:



```
Node:  1 Sector:  3 connects to Node:  0 Sector:  3
Node:  2 Sector:  2 connects to Node:  1 Sector:  1
Node:  0 Sector:  3 connects to Node:  1 Sector:  3
Node:  1 Sector:  1 connects to Node:  2 Sector:  2
Total connections:  4 Dual connections 4
-----------
Converged
```

Figure 80: Result of the test

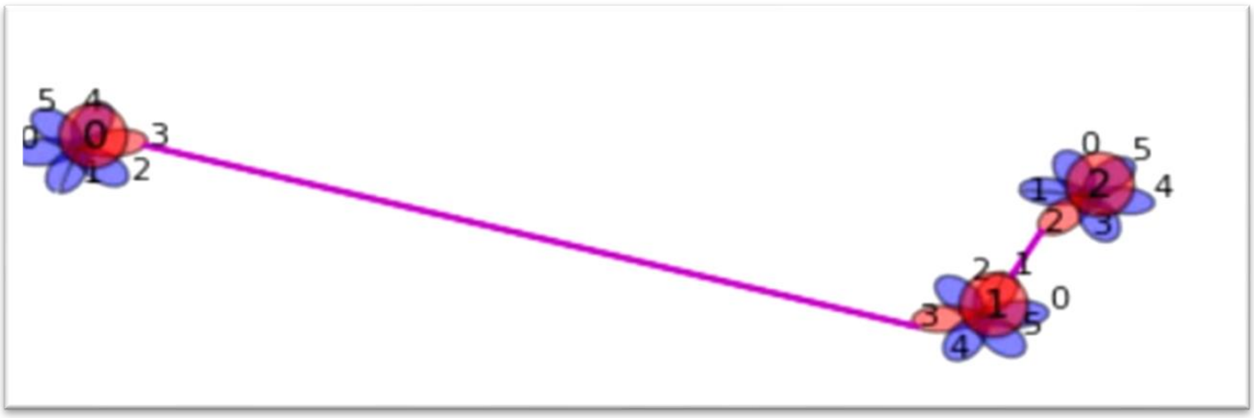


Figure 11: Resulting topology of the test

IMPROVEMENTS BASED ON CLIENT’S FEEDBACK

After all the testing was finished and it seemed as the requirements were met, there is always a difference between the client’s vision and what have actually been done. During weekly meetings, the feedback was given by a client regarding the implementation of the product. Some features had to be altered or removed, due to the difference between the simulated world and the real-world. The example is presented on the Figure 12.

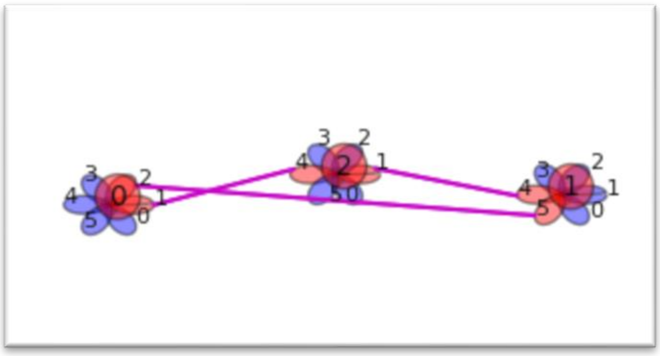


Figure 12: Graph with crossing paths

As it can be seen, the paths (0,1,2,4) and (0,2,1,5) are crossed. It is not supposed to occur in the real-life scenario, because the strength between antennas that are directed towards each other is stronger than the ones that are looking in a bit of different direction. So, the paths are supposed to be (0,2,2,4) and (0,1,1,5).

A client proposed a new rule to solve the problem, which was:

"Hub removes 'weak paths' as follows:

- *The strongest own sector is chosen per peer node (for all j)*
- *The strongest peer sector is chosen per own sector (for all k)"*

After integrating it to the project, the problem has disappeared. Resulting into the topology, that can be seen on the Figure 13.

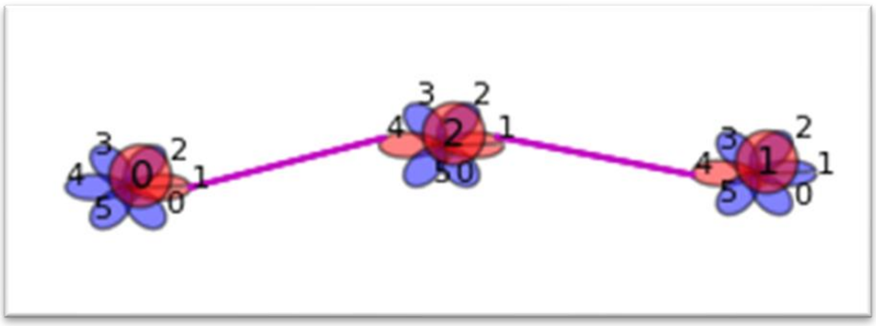


Figure 13: Removing of crossing paths

FURTHER IMPROVEMENTS

Looking at the final product, it is clear that while most requirements have been met, there are still a few possible improvements. Even though there was unfortunately no time to implement everything, it is still important to reflect and discuss what could be improved, and what value that would bring. There are five main improvement points: code optimisations, improvements to link management, the frequency set, improvements to the simulation and visualization.

CODE OPTIMISATION

The first point of improvement in the project is to optimise the code. This could be done in many ways, and many things should be taken into consideration. The first obvious improvement has to do with a design decision that was taken early in the development process, but by the end of the project it was too late to make such a drastic change to the codebase. That decision was to use a list to represent a Radio Path or a Valid Path. After it was too late, it became clear that using a tuple would have been a much better decision. There are two main reasons for that, the first is that the size of those lists never change, which means that using a tuple would have been quite easy to use, if not easier. The second is because lists are a more complex data structure by nature, and by using them instead of tuples, a lot of computational overhead has been unnecessarily introduced.

Furthermore, it is worth considering that the project was implemented fully in *Python*. However, it was not considered whether this program will be able to be compiled and run on the actual hardware. As mentioned, the nodes employ the use of single-board computers with quite limited resources. So, it would have been useful to consider, if needed, how easy it would be to convert this *Python* program into some other, less resource intensive language, such as *C*. Using *C* as the main language was considered but decided against for reasons that have been previously discussed. Alternatively, another plan of approach would be to optimise the code enough that the overhead from using *Python* is almost negligible.

LINK MANAGEMENT

The second point of improvement would be the link management component. Even though it is both correct and complete, it has a higher time and space complexity than preferred. It is particularly important to note that it is not clear whether a more efficient implementation to link management exists, but an assumption. For example, when looking at the code one can see that there is a lot of deep copying, which is quite inefficient both in terms of time and space complexity. It could be argued that there are much more efficient ways to implement link management such that it would utilise less resources.

FREQUANCY SET

The penultimate point of improvement is quite an important one - the frequency set. This one is a bit different than the other points discussed here, as it is not a point of improvement as much as a missing feature from the final product. In the context of the simulation, the frequency set is a feature that could be left out without causing issues since it is concerned with the physical layer, which the simulation abstracts away. However, if the program were to be run on the real hardware, there would be a lot of interference, since the different channel frequencies would clash. It is worth noting that this feature has been partially implemented but was not integrated into the final product due to time constraint.

SIMULATION

Finally, the last key point of improvement is one that would facilitate the move from the simulation into a real-life context, and that is the simulation itself. While the simulation provides basic information about the world, it is far from representative of the discrepancies of the real world. For example, an important aspect that could strongly influence the output of the program is the strengths of the different connections and Radio Paths. In the real world, those values would be jittering quite a lot, where the values would fluctuate by a certain amount between each iteration, even if nodes are not rotating or translating. This is not the case in the simulation. Thus, it could be argued that by implementing random fluctuation of the values, it would bring the simulation to a closer representation of reality.

Another such improvement is about the timing at which the different nodes send and receive data. In the simulation, since the world is centralised and managed by a single program, each node receives data at the same time, and thus sends their data at the same time. This means that they are all running on the same clock, which removes a layer of complexity to the algorithm. A layer that was not taken into consideration. In a real-world application, each node would run on its own clock, and there would be discrepancies between the times at which nodes receive and send data. To solve that problem, it would have been possible to introduce some randomness to the moment in time at which the simulation sends data to each node. As such, we would bring the simulation one more step closer to reality.

VISUALIZATION

There is one last improvement that could have been made, not to the simulation itself, but the visualisation aspect of it. Currently, the simulation visualises each node and its sectors, and between which node-sector pairs there exists a connection. However, it does not show additional information that would have been useful during implementation and testing. Specifically, it does not show which connections were discarded in favour of a stronger 2-hop connection. In other words, it does not visualise the effect link management has on the algorithm. Having this would have proven especially useful, as it would provide a much easier and more intuitive way to see the effect of link management, as opposed to writing down each node’s Valid Paths and Radio Paths and figuring out from there which connections have been discarded.

EVALUATION

This chapter will summarize the entire progress of the project by evaluating the work, that has been done. It will tackle important aspects such a question if all the set goals were achieved and which ones were dropped. Also, it will mention some obstacle that had to be overcome to arrive to the working end product.

OVERALL PROGRESS

This project's initial goal was to re-implement the DND component, and make sure that it is functioning with the hardware that was provided. However, shortly after development had begun, it became clear that testing the product would prove to be difficult using the hardware. The goal was then modified to make sure that the product functions as expected when given synthetic data from a simulation. The product (DND) consists of three core components, pVP-conditioning, path management, and link management. The goal was to implement all three.

When looking at the overall progress throughout the 10 weeks, one could argue that most of the product's features and functionality has been implemented. Although the first few weeks were used solely to gain a better understanding of the project and to conceptualise the approach to be taken, the first core component, pVP-conditioning, was implemented early in the development phase.

In the next two weeks, the focus has shifted towards implementing path management. This proved harder than initially thought but was nevertheless managed. One of the main reasons path managements proved to be harder than initially thought turned out to be an incorrect and incomplete implementation of the pVP-conditioning component. This was due to a misunderstanding of the technical specification of the component. Unfortunately, throughout the project, the latter component had to be revised more than once, after it being found as the culprit to the malfunctions encountered. After the first revision of pVP-conditioning, however, it was concluded that path management was done.

Finally, the final step of the project was to implement link management. Until the very last week, it was apparent that there would be no time to fully implement the final core component. There were too many bugs and things going wrong, and it was important to start working on other things, such as the final report and the reflection components. However, after weeks of trying to fix link management, everything suddenly fell into place. After contacting the product owner, and showing them the final, working version, it was confirmed that link management has been implemented.

Throughout the entire project, a visualisation of the generated graphs was used in order to facilitate testing and implementation. While the simulation provided included a visualisation component, it was implemented in *MATLAB*, meaning it had to be reimplemented in *Python*. This has taken quite some time in order to implement the entire visualisation component from scratch, but it was arguably very useful. Without a proper visualisation tool, it could be strongly argued that there would have been no time to implement link management.

HOW MUCH OF THE INITIAL SCOPE WAS COVERED?

The initial scope of the project was actually much broader than the final scope. Of course, the scope has been changed multiple times in association with the product owner, as the difficulties of the initial scope became clearer as time went on.

The initial scope was then to implement all three components of a node: the DND, the ND (Neighbourhood Discovery), and CM (Connection Management). Furthermore, it was expected that this would be tested and working on a real, physical node that was available in the lab room. In retrospect, it is obvious why this was simply unachievable in 10 weeks' time; there is too much software implementation to be done, and that is besides the hardware difficulties that would have inevitably come up when the testing phase came about.

Using the physical hardware to test the implementation would prove to be too hard for many reasons. The first of those is simply that the hardware is unfamiliar, and it would take arguably too much time to get familiar and proficient with the hardware. Second, the hardware was outdated, which implies more overhead in working out any possible compatibility issues. Furthermore, since the single-board computers being used are quite resource limited, that would introduce one more layer of complexity; optimising the implementation such that it would run on limited resources. And finally, the biggest issue was that testing the implementation would be extremely hard. Since the node's communicate using directed Wi-Fi antennas, having multiple sectors from multiple nodes running and broadcasting data in the lab room would be catastrophic. This means that the nodes would have to be installed quite a distance away from each other, which would be troublesome.

Thus, the product owner proposed to use a simulation that could be used to inject synthetic data into each node's algorithms. This would allow the abstraction of the data layer away and help focus the attention into the software aspects of the product. However, it also became clear that most of the work that needed to be done was in the DND component, since the ND component was already (partially) implemented Furthermore, a big part of CM had been abstracted away with the physical layer, since CM was responsible for managing its sector's connections, both on a logical and physical level.

And so, the initial scope has effectively been cut down and concentrated on the biggest software component of a node, the DND. From the DND, the final product of the project managed to include and meet all the requirements that were specified, with one unfortunate exception: frequency management. That is, managing on which frequency a certain connection should use, such that there is no interference between the different connections.

The reason this functionality was not fully implemented is two-fold. The first is simply due to time constraints, while the second is because it was unclear how such a feature fit into DND. There was some ambiguity about whether frequency management is part of DND or CM. Furthermore, it proved quite difficult to implement frequency management given the code structure and design of the project. It was not clear how the simulation would handle frequency data, and the possible solutions implied refactoring code, for which there was unfortunately no time.

HOW THE PROJECT BENEFITED FROM THE SIMULATION?

As discussed above, the simulation allowed for the physical layer to be abstracted away. Instead, the project's scope and focus became on the software and business logic of the DND algorithm. This allowed the attention and work to be put in implementing a functionally complete and correct algorithm, while minimising the space for errors and issues. Not only did the simulation eliminate points of failure, but it also allowed control over the data that is being injected into the DND algorithm.

By controlling the DND's input, it became almost trivial to test a certain situation or case that would normally cause issues. It was possible to specify the number of nodes in the simulation, how many sectors they have, their initial positions, their initial orientations, and even their directional and rotational velocities. This allowed for a controlled testing environment, where specific cases could be used to test specific aspects of the algorithm. For example, a certain case that was tested was to have multiple static nodes, and one of them would steadily move away from them. Next to that, we would have another case where the node would start far away, and steadily move close to the other static nodes. These two situations were used to test the ad-hoc nature of the algorithm. There were many other cases that were used to test many other features of the algorithm, including handovers, and 2-hop optimisations.

Finally, one of the biggest advantages of using a simulation was, as previously discussed, the existence of the visualiser. Having a visualisation of the nodes, which sectors from which nodes are active, and which nodes are connected to which nodes, proved to be immensely useful in development. The main reason for that is because it became much easier to spot errors and when things didn't go as expected, as opposed to if the algorithm had been run on the physical node hardware.

Even under the assumption that the algorithm works without any problems on the physical hardware, it would be a very laborious process to come up with a graph representing the different nodes and their connections to their neighbours. Each node's data would have to be recorded separately, and then combine them in order to gain an eagle-eyed view of the system, and then finally draw the graph by hand. One can imagine how much easier it is to simply have the simulation do all of this on a single machine and output the graph automatically.