UCL Université catholique de Louvain

École polytechnique de Louvain (EPL)

epl

Temporary infrastructures and Ad-Hoc networking for Mars explorers

Dissertation presented by
Maxime DE STREEL

for obtaining the Master's degree in
Computer Science Engineering
Option(s): Network Security

Supervisor(s)
Etienne RIVIÈRE, Raziel CARVAJAL-GÓMEZ , Michael SAINT-GUILLAIN

Reader(s)
Ramin SADRE, Ayham KASSAB

Academic year 2019-2020

Abstract

In this thesis, we present a solution to improve communication for astronauts on extra-planetary expeditions. Our approach relies on the use of a temporary infrastructure and Ad-Hoc networking. The basic idea of the design is to dynamically position infrastructure nodes along their path to maintain a multi-hop path between the base station and the astronauts. Knowing when to place a temporary infrastructure is a challenge; this thesis proposes a solution to this with our backbone placement protocol. Having infrastructure nodes in an Ad-Hoc network amounts to a new sort of routing protocol for Ad-Hoc networks. We have investigated modifications to the basic PRoPHET routing protocol [1], in order to make it suitable for a network with temporary fixed infrastructure nodes. Our modifications address the low latency of PRoPHET by taking advantage of the fixed infrastructure nodes to quickly route messages. The proposed solution will be tested on hardware and real conditions during a real case scenario at the Mars Desert Research Station in the Utah desert. Unfortunately the simulation had to be postponed to mars 2021 due to the actual circumstances. In the mean time the proposed solutions have been evaluated with the omnet++ network simulator.

Acknowledgements

Foremost, I would like to express my sincere gratitude not only to my advisor Prof. Etienne Riviere but also to Raziel Carvajal Gomez for there patience, motivation and immense knowledge sharing. Their guidance helped me in all time of research and writing of this thesis. I could not have imagined having better advisors for my master thesis.

I also would like to thank Michael Saint-Guillain for his support in the organisation of the UCL to Mars project as for his guidance for the writing of this thesis.

Finally, I would like to thank my parents for their wise counsel and sympathetic ear. You are always there for me. Finally, I could not have completed this dissertation without the support of my girlfriend, Audrey Desclée, who provided stimulating discussions as well as happy distractions to rest my mind outside of my research.

Contents

Chapter 1

Introduction

The interest for the red planet has gained a lot of attention in the past years, the space conquest is back. For the past three years 28 missions towards Mars have been launched [2]. Elon Musk, CEO of SpaceX[1], recently published that his Interplanetary Transport System (ITS) would be ready for 2024[2]. The first astronauts on Mars will need to be able to communicate between them. In comparison to places on Earth, there is no 4G on Mars or any other communication technology. This thesis gives an answer to the following question: How to maintain communication between astronauts that are continuously moving in a place with no communication infrastructure?

In order to simulate the martian surface, there are places on Earth that are very similar to Mars. One of those places is the Utah desert. Each year a team from the Université Catholique de Louvain (UCLouvain) is organising a two week simulation in the Utah desert to allow students experience the living conditions on the red planet.

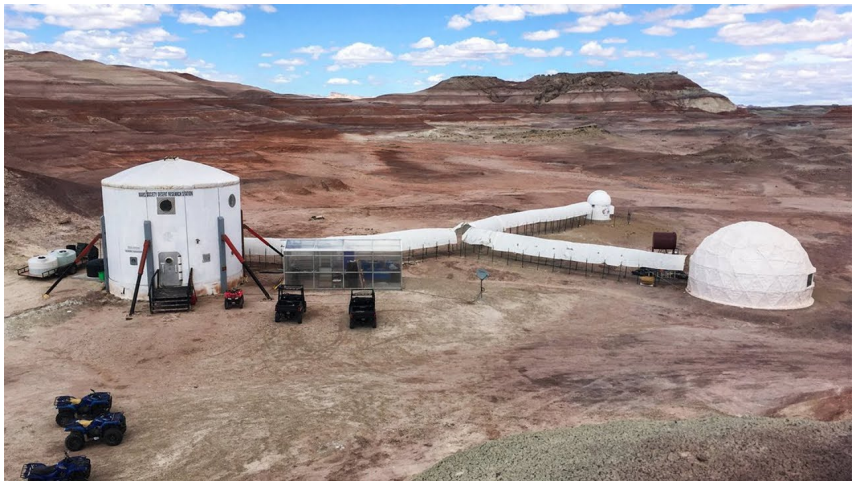


Figure 1.1: Base station in the desert of Utah, USA

1.1 UCL to Mars

UCL to Mars is a non-profit organisation that select researchers and students from the UCLouvain [3] to visit the Mars Desert Research Station [4] (MDRS) in the desert of Utah, USA to simulate missions on Mars. The UCL-to-Mars team conducts scientific experiments [5] and performs geo-

[1] https://www.spacex.com
[2] https://www.nationalgeographic.fr/espace/elon-musk-dans-7-ans-spacex-pourra-envoyer-des-hommes-sur-mars
[3] www.uclouvain.be
[4] mdrs.marssociety.org
[5] www.ucltomars.org/our-experiments

logical fieldwork. The MDRS is in fact a laboratory similar to an extraterrestrial station where its guests: live in a confined space, perform extravehicular activities, wear space suits and eat lyophilized food. Figure 1.1 depicts this laboratory where its *astronauts* (researches and students) spend most of their time. It is composed of a main hub (on the left), a scientific dome (on the right) and the observatory (at the back). The main dome serves as canteen, dormitories and working place. The program UCL to Mars receives the support of the Mars Society [6], an American organisation that promotes exploration and colonisation of Mars.

Figure 1.2: Astronauts performing a soil exploration

Astronauts also perform soil explorations wearing space suits as shown in Figure 1.2. These explorations might result in long walks away from the base station as the following example describes. Each astronaut possesses a built-in ad-hoc wireless device as part of its space suit. They leave the base station and continue their exploration in small groups or individually. Once astronauts have completed their job, they come back to the base station using the same way as they went out.

It is thus vital to maintain reliable communication between astronauts and the base station. In case of an emergency, for example, the base station should be notified to be able to act accordingly; informing nearby astronauts or sending a rescue team.

1.2 Problem statement

The establishment of a reliable infrastructure of communication in the context of extraterrestrial soil explorations is a challenging task. Such infrastructure must be built on-demand because the itinerary of explorations might change every day. Resiliency to interferences should also be a requirement given that materials in unexplored soils might disrupt radio frequencies. The area of communication of this infrastructure should cover every astronaut during the entire duration of every exploration while maintaining an open channel to contact the base station.

The solution proposed in this thesis is based on different assumptions. In wireless protocols it is often assumed that nodes are capable to determine the distance at which a given message was send. This is of course an important assumption and we will exploit it in this thesis. In Chapter 3 we will validate this assumption experimentally.

[6] www.marssociety.org

1.3 Hybrid approach for extraterrestrial explorations

The aim of the present research work is to provide a reliable infrastructure of communication for extraterrestrial soil explorations in the context of the UCL to Mars program. The main contribution in this work is to combine the capabilities of ad-hoc networks along with fixed access points that will be placed by astronauts while they follow their itineraries of explorations.

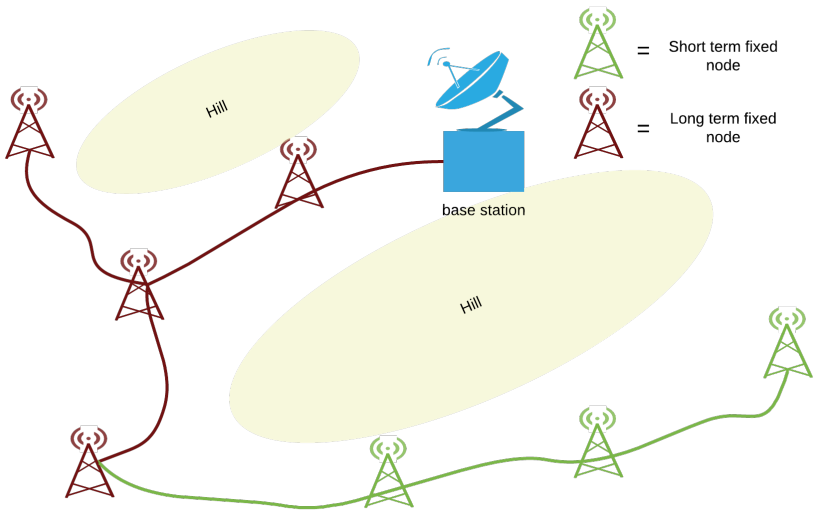


Figure 1.3: Area surrounding the MDRS

Figure 1.3 depicts an example of the proposed approach where ad-hoc devices (hold by astronauts) might interoperate with a temporarily infrastructure to maintain a reliable communication with the MDRS station. The main idea is to let astronauts set up fixed nodes forming a temporarily infrastructure, this will complement the capabilities of ad-hoc devices allowing to extend the range of communication during soil explorations.

In green, you can see an example of a path astronauts might follow in an outside mission. Most of the tasks are developed on the main path (brown). Long term and short term fixed nodes differentiate themselves on their batteries. Long term fixed nodes are meant to stay several days and are placed on the main path. Short term fixed nodes are taken back after the outside activity and are placed between the main path and the target. Thanks to this there will always be a multi-hop path between the astronauts and the base station. This route is composed of long and short term fixed nodes.

Generally speaking this thesis contains two main contributions: i) a proof of concept of the proposed hybrid approach to aid extraterrestrial explorations that have been tested via simulations and ii) the deployment of the proposed approach on wireless devices (Raspberry PIs) to mimic a soil exploration.

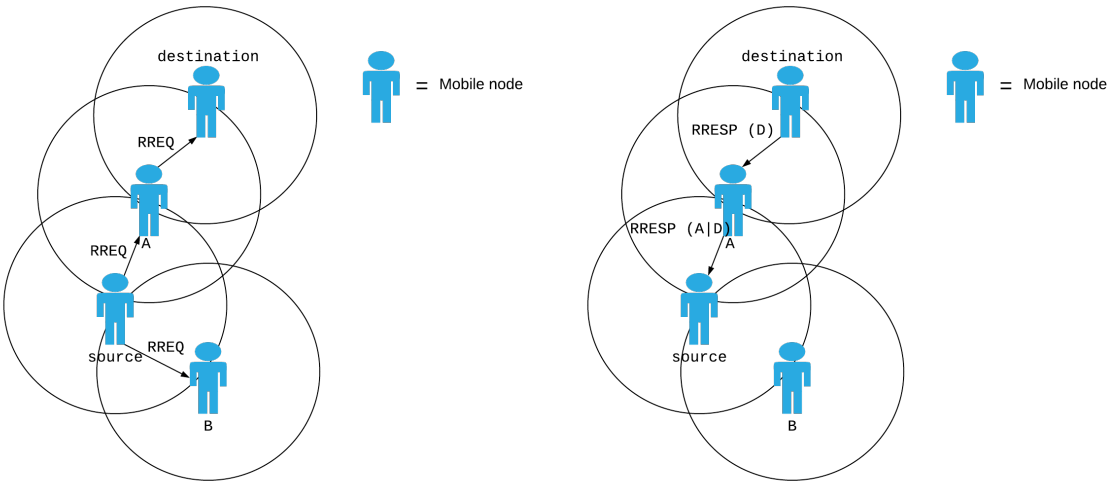
Chapter 2

Background

This section covers the technical background necessary to have a good understanding of the contributions in this thesis. We will shortly describe what Mobile Ad-Hoc networks (MANETs) are and how they work. Then, we will explain what Delay Tolerant Networks (DTNs) are and their main main advantages and drawbacks. Finally we will cover routing in DTNs with the PRoPHET protocol and the GTMX forwarding strategy.

2.1 Mobile Ad-Hoc networks (MANETs)

Mobile Ad-Hoc networks (MANETs) are composed of a set of wireless devices (laptop, cell phones, PDA, etc...) able to spontaneously interconnect without any pre-existing infrastructure, such as fixed wireless routers. For example, a MANET deployment might emerge using a group of cellphones. To reach distant participants in the network, every node follows a multi-hop approach to route messages. A recent protest in the city of Hong Kong [1] shows how big such networks can be. This example shows how MANETs can be useful in cases were the government blocks the network but they can also be very useful in regions where no network is available; for instance, after a natural disaster.



(a) Route request

(b) Route response

Figure 2.1: Route discovery in MANETs

MANETs are self-configuring wireless networks of mobile nodes. Each node is able to move at will, meaning that they can leave the network at any time, creating arbitrary network typologies.

[1] https://www.forbes.com/sites/johnkoetsier/2019/09/02/hong-kong-protestors-using-mesh-messaging-app-china-cant-block-usag

To route messages between a pair of nodes, a route between a source and a destination has to be found. Figure 2.1 shows how a route is discovered. First the source node is broadcasting a route request for the destination node, every node broadcast that same request. When the destination node receives the route request it answers with a route response. Every intermediary node in the discovered path adds his node identifier to the response. Once the source node hears from a new route, it starts sending messages to the destination.

The operating conditions of MANETs are challenging: nodes are battery powered with a single processor and few RAM, every node possesses a limited range of communication and nodes move at will. Indeed, having several devices communicating with each other will lead to interferences and thus packet losses. Furthermore, with few devices in the network there might be more disruptions in letting two nodes communicate with each other because it is more likely that they will be out of range.

Besides the mobility of nodes MANETs support basic primitives of communication, such as broadcast, unicast-cast and routing. For a broadcast routing, no path need to be found since every node is a destination. For unicast routing, the source node need to find a route to the destination (as explained above). The main protocols for routing in connected networks are Ad-Hoc On-Demande Distance Vector (AODV) [3], Destination Sequenced Distance Vector (DSDV) [4], Optimized Link State Rouring Protocol (OLSR) [5]. Unfortunately they do not work in partitioned networks since they need a path between the source and destination node at the time of sending. An alternative approach to route messages through partitioned networks is the use of Delay Tolerant Networks (DTNs).

2.2 Delay Tolerant Network (DTN)

The Delay Tolerant Network architecture has been designed to deal with challenged networks (such as MANETs, vehicular had-hoc networks, satelite communication, etc.) with the following characteristics:

- **Intermittent connectivity.** Connection links are transient and short in time due to mobility;
- **Long or variable latency**;
- **High error rates.** Bit errors on links require correction or re-transmission of packets.

Delay tolerant networks (DTN) is an architecture initially proposed by Kevin Fall [6] to let challenged networks interoperate with each other. DTNs work above the existing protocol stacks in various network architectures and provide a store-and-forward gateway function between them.

In the context of Inter Planetary Networks (IPNs), DTNs enable reliable communication between satellites and base stations. This network architecture leverages reliability of routing between nodes in the presence of high latency, which is a constant in spatial communications. In case a message can not reach its destination, for instance, this message is hold by a node until a new route is available as depicted in Figure 2.2. Messages are stored and then forwarded. They will be removed from the buffer when their lifetime expires or when it is impossible to store more messages. This approach can be used in IPNs but also to Vehicular networks and extra-planetary explorations, taking advantage of their high degrees of mobility [6, 7]. An implementation of the DTN architecture have been developed by the Delay Tolerant Network Research Group (DTNRG) [8] to ensure network interoperability. In this context, interoperability refers to integrate several network approaches, for instance, a vehicular network could then use the IPN infrastructure for routing.

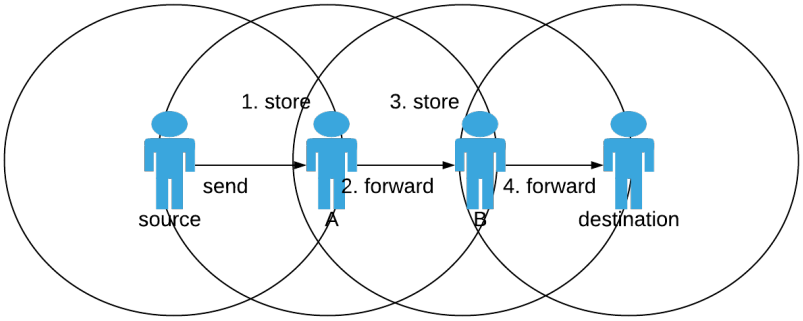
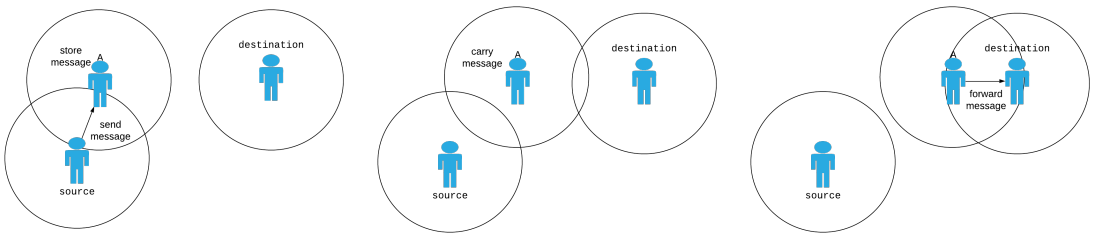


Figure 2.2: Store-and-forward approach.

2.2.1 Store-and-forward principle

As the name suggests, the store-and-forward principle let nodes to store messages and forward them to other neighbouring peers based on a certain strategy. Each time a node receives a message, this node stores such message and carries it for a predefined time. The store-and-forward principle allows a message to be carried and delivered to a destination node even if no path exists when such message have been sent by a source node.

Figure 2.3 shows an example where initially there is no path between two nodes, a source and a destination. Node A stores a copy of the message that have been sent by the source. Since nodes are mobile this store-and-forward approach allows messages to be forwarded even if there was no initial path between the source and the destination. Indeed node A has stored the message and is then moving around, after some time this node encounters the destination node and deliver the message to it.



(a) store message (b) carry message (c) forward message

Figure 2.3: Store Carry Forward example

2.3 Routing in DTNs

The present research work intents to use a store-and-forward approach in the context of a MANET deployment. The network suffers from frequent connectivity disruptions, making the topology only intermittently and partially connected. This means that there is a very low probability that an end-to-end path exists between a given pair of nodes at a given time. End-to-end paths can exists temporarily, or may sometimes never exists [9, 10]. Routing is a challenging issue in DTNs, as mentioned by Jain at al. [11]. Due to the operating condition in disconnected MANETs routing algorithms such as OLSR, which is based on the spreading of control information, or AODV, that is an on-demand routing protocol, experience high error rates and low throughput of messages.

A possible solution when nothing is known about the behaviour of nodes could be Epidemic routing. This routing approach is flooding-based in nature, in the most simple case, epidemic routing is pure flooding. Since Epidemic routing can lead to high rate of collisions and contentions in the network, one would prefer to limit message duplication and instead use a routing heuristic. The DTN architecture defines different types of heuristics: scheduled, opportunistic, and predicted [8]. *Scheduled* contacts can be used, for example, between a base station on Earth and an orbiting relay satellite. *Opportunistic* contacts are created by the presence of two entities at the same place, the meeting was neither scheduled nor predicted. Finally, *predicted* contacts are made by analysing previous observations.

The present work focuses on the last heuristic, predicted contacts, in the context of soil explorations movements of astronauts are predictable (for instance, following a specific path at a fixed velocity). A common routing protocol in delay tolerant networks is the Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) [1]. We chose the PRoPHET routing protocol over the other approaches since Ad-Hoc nodes will have very predictable movements, which will increase the performances of PRoPHET since routing in this approach is based on the probability that two Ad-Hoc nodes will encounter.

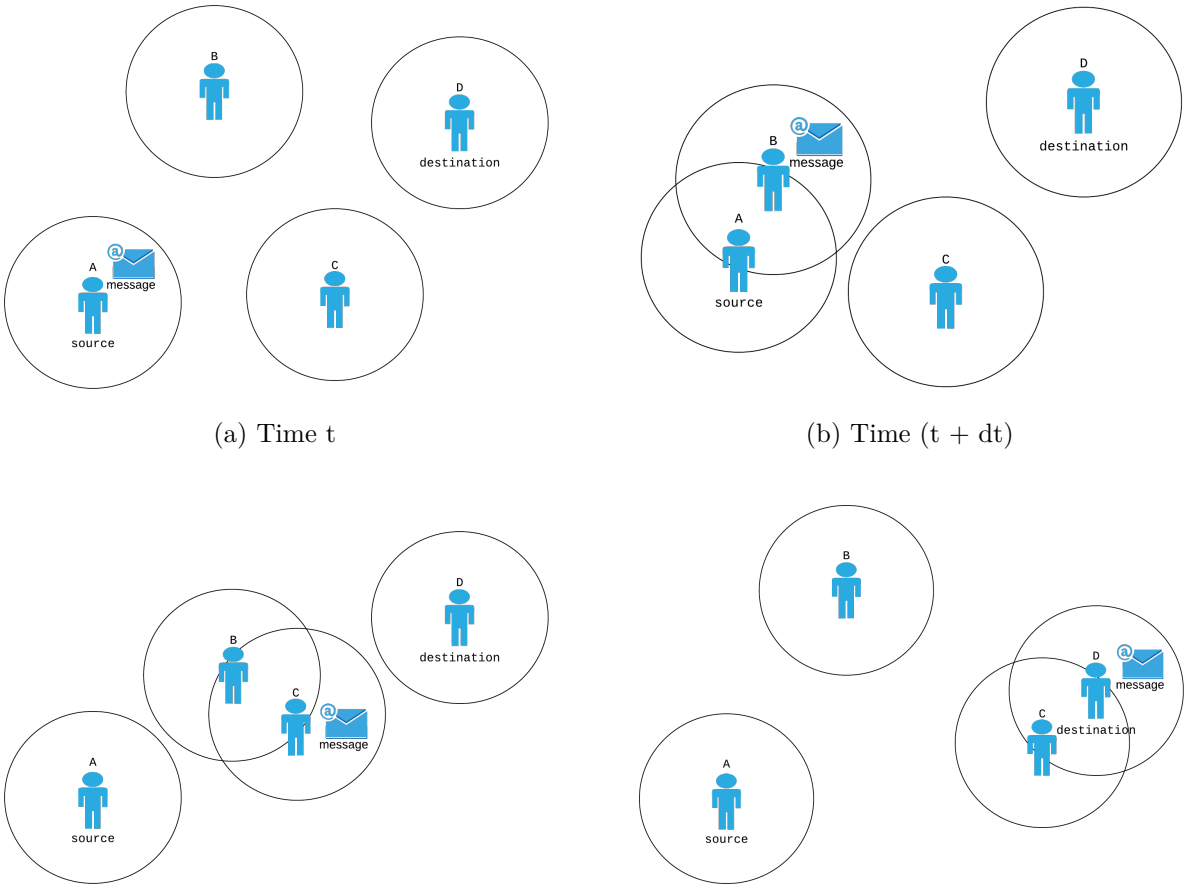


Figure 2.4: Example of transitive communication

2.3.1 PRoPHET

The Probabilistic Routing Protocol using History of Encounters and Transitivity (PRoPHET) [1] enables communication between nodes in an intermittently connected network where at least some of the nodes are mobile. This protocol relies on the store-and-forward principle to deal

with mobility in order to deliver messages between a pair of nodes as depicted in Figure 2.4. Node A sends a message to node D. Given that D is out of the communication range of A, this node transmit the message to node B (in this example, node B has previously met node C that met node D). B is moving around and when it encounters C it will forward the message since this latter node has more chance to see node D. Finally, node C encounters D and forward the message originally sent by A.

Delivery Predictability Map

In PRoPHET, the delivery predictability map is a data structure that stores probabilities of estimation to encounter another node. Each node maintains its predictability map up to date. Every time nodes encounter another neighbouring peer, they calculate (2.3.1) his new predictability based on information received by the encountered node. The delivery predictability map is in fact a history of encounters. All probabilities are only valid for a certain time. After that given time, the node identifier is erased from the predictability map (or the corresponding probability is set to 0). Table 2.1 shows an example of a delivery predictability map for Node B, following the example in Figure 2.4.

Node EID	Probability
Node A	0.765
Node B	1
Node C	0.632
Node D	0.364

Table 2.1: Delivery Predictability Map example

As the example shows, the probability to encounter B (itself) is set to 1. One could ask why bothering to set his own probability into the map since we will never forward a message to ourselves; this serves to simplify the probability of encounters of other nodes. When two nodes exchange their maps, the updated algorithm of the map do not require a node identifier, a receiver node only need its own delivery predictability map.

Delivery Predictability Calculation

When two nodes encounter they first exchange their delivery predictability maps. They then update their own map based on the one they just received. The predictability is then calculated with Equation 2.1 where δ is a small positive number that serves as upper bound for $P_{(A,B)}$. On the first encounter, the delivery predictability is set to a default value $p_{\text{first\_encounter}}$. Since PRoPHET has no prior knowledge about whether is will be a unique encounter or if they will encounter frequently, $p_{\text{first\_encounter}}$ is set to 0.5.

$$P_{(A,B)} = P_{(A,B)_{\text{old}}} + (1 - \delta - P_{(A,B)_{\text{old}}}) * P_{\text{encounter}} \tag{2.1}$$

In some cases it is possible that a single encounter in terms of proximity happens but that this results in multiple communication opportunities closely spaced in time. In those cases, it is not appropriate to increase the probability by the same amount. To reduce the distortion of the delivery predictability in these cases, $P_{\text{encounter}}$ is a function of the interval since the last encounter resulted in an update of the predictability map. The corresponding curve of thi function is shown in Figure 2.5.

The delivery predictability also has a transitive property, that is, if a node A frequently encounter node B and that node B frequently encounters node C, there is a probability that node C will encounter node A (itself or through node B). Node C is thus a good node to forward messages to that has as destination node A. Equation 2.2 shows how this transitivity affects the

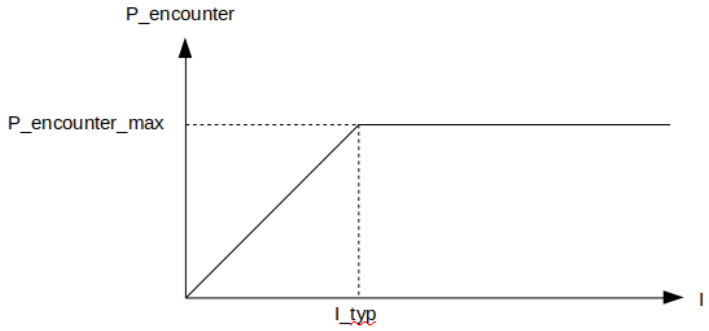


Figure 2.5: $P_{\text{encounter}}$ as function of time interval, I, between updates

delivery predictability, such that $P_{(B,C)_{\text{recv}}}$ is the value from the encounter with a node and $0 \leq \beta \leq 1$ is a constant that controls how large the impact of the transitivity should be on the delivery predictability.

$$P_{(A,C)} = \text{MAX}(P_{(A,C)_{\text{old}}}, P_{(A,B)} * P_{(B,C)_{\text{recv}}} * \beta) \tag{2.2}$$

Furthermore, if a pair of nodes do not encounter during an interval, they are less likely to be good forwarders of message for each other thus the delivery predictabilities value must age. The second part of the updates of the metric values is the application of the aging function shown with Equation 2.3, where K is the time elapsed between the last time the metric value has been aged and $0 \leq \gamma \leq 1$ which is the aging constant.

$$P_{(A,B)} = P_{(A,B)_{\text{old}}} * \gamma^K \tag{2.3}$$

Forwarding strategy

Once two nodes encounter, they first exchange their delivery predictability maps. Once they have updated their own delivery predictabilities with the one they just received, they will look for some messages to forward to the encountered node. When and to whom a message should be forwarded is decided by the forwarding strategy.

Different forwarding strategies for PRoPHET routing already exist [1]. We will focus on the GTMX forwarding strategy. The GTMX forwarding strategy described in Algorithm 1 is a copy of the algorithm described in [1] and works as following: if the encountered node has a greater probability to encounter the destination node (line 18), the message is forwarded to the encountered node. At the other hand, if the destination is not in our delivery predictability map, forward the message (line 16, 17). The last reason is motivated by the fact that all nodes may not support the PRoPHET routing protocol. Furthermore, each message is forwarded at most NF_MAX times to other nodes apart from the destination (line 18).

The base routing strategy used by PRoPHET is also described in Algorithm 1. When a PRoPHET node receives a DeliveryPredictabilityMap (line 4) from another node it will first apply the aging function (Equation 2.3) on their local delivery predictability map and update it according Equation 2.1 (line 5, 6). After that, the node will go through all his stored messages and find those which he can forward to the encountered node. A message should be forwarded if the destination node is the encountered node or if the forwarding strategy returns true (line 8).

Algorithm 1 Forwarding strategy

```
1: variables
2:     dpm: DeliveryPredictabilityMap, init: empty          ▷ the local delivery predictabilities
3:     strategy: GTMX                                       ▷ the forwarding strategy, here GTMX is used
4: procedure PRoPHET.receive(DeliveryPredictabilityMap ndpm)
5:     dpm.age()                                            ▷ age the metric values
6:     dpm.update(ndpm)                                     ▷ update the local metric values with neighbor dpm
7:     m_i ← messages[i]
8:     if m_i.getDestination() == ndpm.getSrc() || strategy.shallForward(m_i, ndpm.getSrc() then
9:         MAC.unicast(m_i, ndpm.getSrc())
```

GTMX forwarding strategy

```
10: constants
11:     NF_max: int                                          ▷ maximum number of forwards
12: variables
13:     NF_map: map, init: empty                             ▷ key: Message, value: number of forwards
14: procedure shallForward(Message m, DeliveryPredictabilityMap ndpm)
15:     NF ← NF_map.get(m)
16:     if !dpm.contains(m.destination()) then
17:         return true
18:     if NF <= NF_max && ndpm.get(m.destination()) > dpm.get(m.destination()) then
19:         NF ← NF + 1
20:         NF_map.set(m, NF)
21:         return true
22:     return false
```

The GTMX forwarding strategy is thus a simple algorithm that routes messages based on the probability that the encountered node has to see the destination node. The probabilities are calculated in a way where it is easy to favour a route by tuning the parameters in Equation 2.2 and Equation 2.1 to increase either our own probabilities our the one we learned from other nodes. In the Mars use case we will have a static infrastructure, a good idea would be to take advantage of it to route messages through the infrastructure. With the GTMX forwarding strategy this can easily be done.

Chapter 3

Configuring a deployment to mimic extraterrestrial explorations

From September 2019 until March 2020, we worked on a real use case evaluation in the Utah desert at the Mars Desert Research Station (MDRS). During that time we configure a deployment of an ad-hoc network using Raspberry PIs (model 3B+). We used the IBR-DTN [12] bundle protocol as framework for our Delay Tolerant Network application. This Chapter will go through the work done during 6 month and that will be used next year during the UCL to Mars 2021 mission.

3.1 The hardware: Raspberry Pi 3B+

The hardware chosen for the experimental use case was the Raspberry pi 3B+[1]. The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV. It's capable of doing everything you had expect a desktop computer to do, from browsing the internet and playing high definition video, to making spreadsheets, word processing and playing games. The Raspberry Pi 3B+ is composed of the Broadcom BCM2837BO, Cortex-A53 (ARMv8) 64-bit Soc @ 1.4GHz cpu and has an 1GB LPDDR2 SDRAM. The Raspberry PI is also equipped of the following WiFi chip, 2.4GHz and 5GHz IEEE 802.11.b/g/n/ac wireless LAN. As for the power input, it supports a 5V/2.5A power input. Figure 3.1 is an image of the Raspberry Pi 3B+, as we can see it has several GPIO pins.

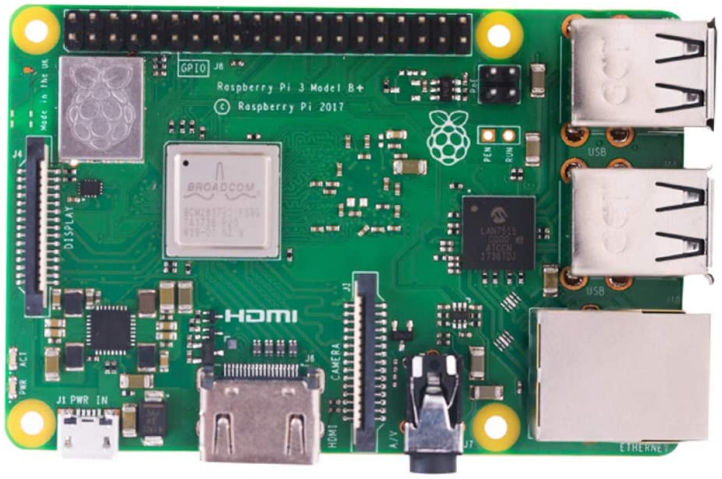


Figure 3.1: Raspberry Pi 3B+

[1] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

We chose the Raspberry Pi 3B+ as hardware for the experimental use case since it is very small, lightweight and does not need a big power supply to work. Since astronauts will wear those the hole time, they will need small and lightweight devices. Another big advantage of the Raspberry pi suit is that it has GPIO pins where we directly can connect some sensors to monitor the astronauts.

3.1.1 Hardware performance

First we had to evaluate the hardware, especially the wifi chip. It is important to understand the limits of the hardware. In our case we only measured the wifi performances since it is the most critical aspect for this thesis. We needed to know what the maximum range was and if it was possible to evaluate the distance at which a given message was send.

To do so, we configured two Raspberry Pi and configured them in Ad-Hoc mode. After that, we installed and created a simple ping application with one Raspberry Pi as receiver and the other as sender. We measured the signal strength when a message arrived and logged the distance at which both Raspberry Pi where. Figure 3.2 shows the results of this experiment.

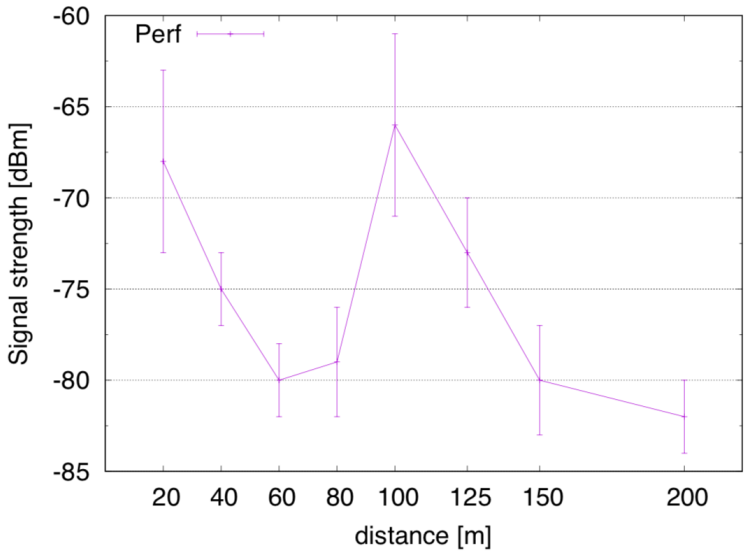


Figure 3.2: Wifi signal strength with wifi power control

As we can see, the measured signal strength on the receiver decreases with respect to the distance until we reach -80 [dBm], then it increases up to -63 [dBm] and finally it decreases again. We didn't understand this behaviour well. After a discussion with a telecommunication expert (Claude Oestges), we understood that this peak is due to the wifi power control. When the sender detects that it is loosing the connection with the peer, it will give more power to maintain the connection. Claude Oestges suggested to disable this functionality on the Raspberry Pi since our main goal was to spare power and to measure the distance at which a message was send.

After disabling the power control of the wifi chip we did the experiment again and obtained the results on Figure 3.3. As we can see, there is no longer a peak at 100 [m], disabling the power control has effectively removed our problem. Analysing the graph on Figure 3.3 we can also say that there is a link between the signal strength and the distance at which the message was send. But, even if messages were well received up to 200 [m] we can see that after 100 [m] it becomes quite difficult to link the signal strength with the distance. Therefore we decided to set

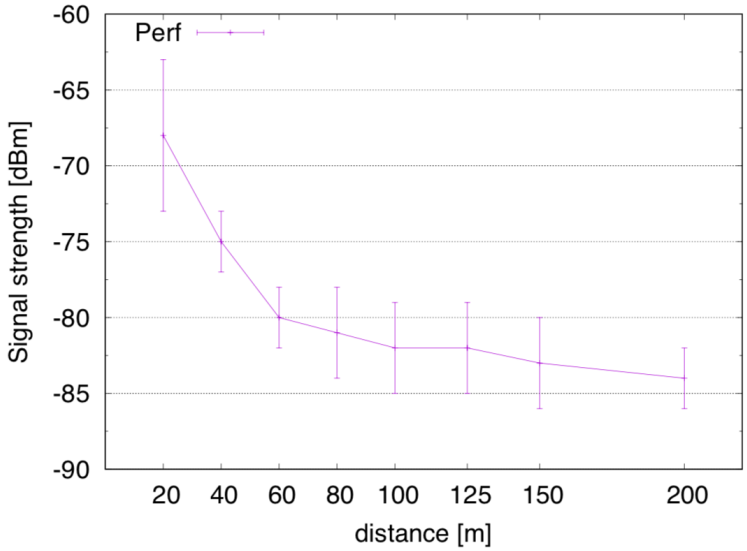


Figure 3.3: Wifi signal strength without wifi power control

the maximum range of a Raspberry Pi to 100 [m].

With this experiment we validated the assumption that a node is capable to determine at which distance a given message was send. Furthermore, the chosen hardware fulfils now all our requirements, which are, small and compact, light, small power supply and a wifi chip that is able to determine the distance at which a message was send.

3.2 IBR-DTN: a lightweight implementation of the bundle protocol

The standard protocol used in DTNs is the Bundle Protocol [13]. Unfortunately the Bundle Protocol reference implementation DTN2 is not suited for Embedded Systems like the Raspberry Pi 3B+. Instead of using DTN2 we used IBR-DTN, which aims to be a fully compliant Bundle Protocol daemon and is designed to run on OpenWRT[2], which is a Linux distribution specifically built to run on embedded hardware. Schildt et al. [12] showed that IBR-DTN fully compatible is with the Bundle Protocol reference implementation (DTN2) and that in most cases its speed supersedes that of DTN2. Taking into account that IBR-DTN is fully interoperable with DTN2, that its performances supersedes thoses of DTN2 and that it is designed for embedded systems, we did not see any reason not to take IBR-DTN as Bundle Protocol implementation.

One of the big advantages of IBR-DTN is that the PRoPHET protocol already implemented is and that it provides a simple but extended socket API interface to communicate with the daemon. To avoid reimplementing the complex bundle streaming protocol in each application, an IBR-DTN library can be linked to applications simplifying the creation of bundles. This approach has the advantage that all supported DTN features of the daemon are immediately available as they can be enabled by setting the appropriate bundle headers.

The documentation of the official IBR-DTN github repository suggest to use the auzias/ibrdtn-api[3] as library. This library is a java library that directly communicate with the socket API

[2]http://openwrt.org
[3]https://github.com/auzias/ibrdtn-api

interface of IBR-DTN daemon. It was simple to use and offered a lot of different configurations but was still missing some important features that we needed. One of them is the time-synchronisation. The normal behaviour of IBR-DTN is to drop packets with an invalid timestamp (i.e. a timestamp from the future), which is of course the expected behaviour in networks where all nodes are synchronised. In the Mars use case, we don't have any internet connection and it is thus not possible to have all nodes synchronised. The IBR-DTN daemon offers the possibility to configure the time synchronisation to false but the ibrdtn-api did not. We had thus to implement this feature in the library. Furthermore, the library was written for java 8 and we needed java 9 for our application to work, we thus upgraded the library to java 9, our version of the ibrdtn-api is available on github[4].

Now that we have been through the hardware and the Bundle Protocol implementation that we will use, we will discuss the DTN application that we developed for the astronauts.

3.3 DTN application

In this section we will discuss the DTN application we developed for the Mars use case. The application monitors the astronaut's sensor data and send it towards the base station. The base station can then, with the User Interface monitor every astronaut on the field. If the astronauts are equipped with screens they will also be able to monitor their data.

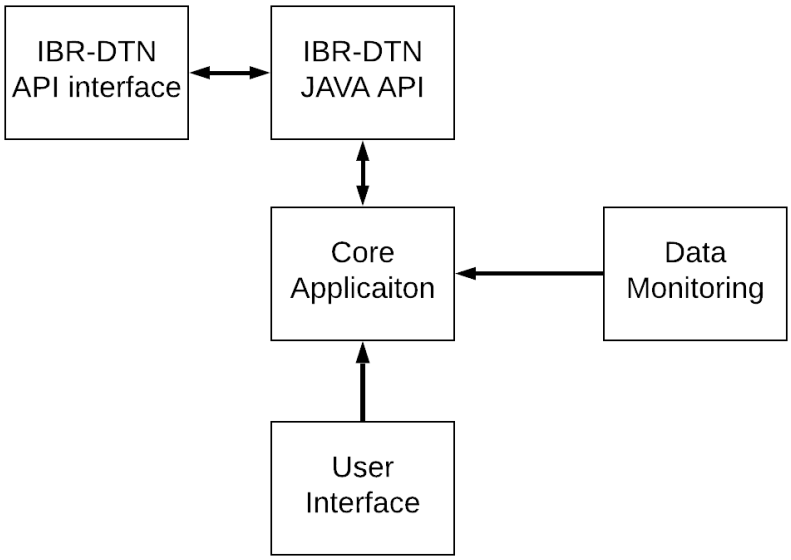


Figure 3.4: DTN application structure

The application works on top of the IBR-DTN daemon and is composed of several components. Figure 3.4 shows how those components interact between each other. First it has the core application, the core application does the communication between the other components, especially between the data monitoring component and the IBR-DTN java API. The data component is a python component that listen to the different sensors connected to the Raspberry Pi and send the data to the core application. Finally we have the User interface that is responsible to show the real time data of each astronaut in a a separate graph.

[4]https://github.com/destmaxi/ibrdtn-api

The DTN application source code is also available on github[5].

3.3.1 Core application

The core application is the heart of the DTN application, it handles the requests from the python data manager, converts the data into bundles and sends them through the java API to the IBR-DTN daemon. It also writes the data to the file system in case the persistent data storage is enabled. The received data is send to the base station but also to every astronaut in range.

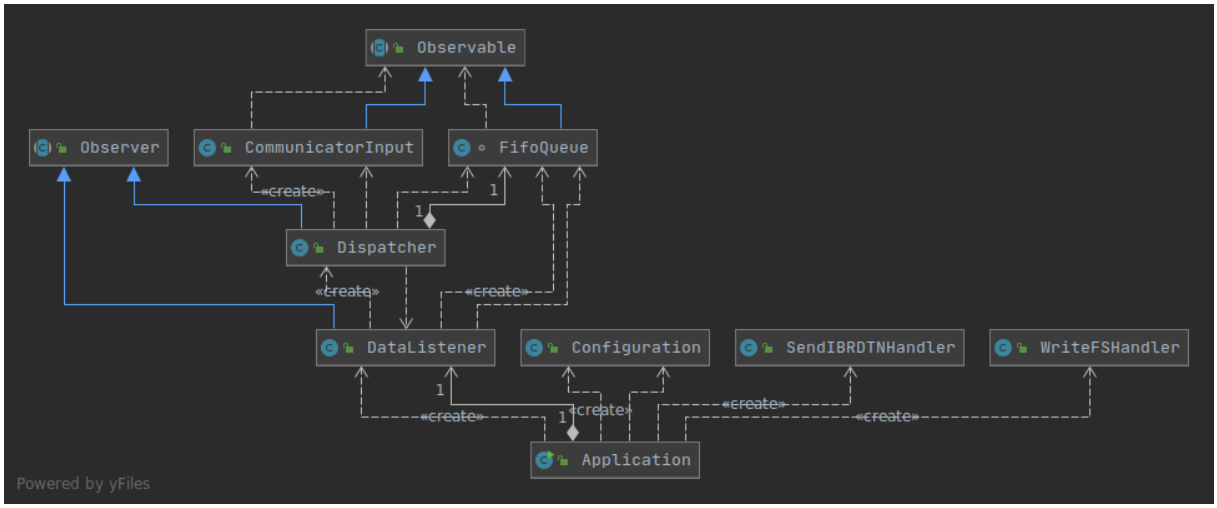


Figure 3.5: Core application structure

Figure 3.5 shows the class diagram of the core application. The configuration class takes a `config.properties` file as input that will set the different parameters. The `CommunicatorInput` class is listening to incoming bundles from the IBR-DTN java API. Whenever a bundle arrives, it will notify his observer, in this case the `DataListener`. The `DataListener` handles the incoming bundles through the `FifoQueue` and stores them if needed. It is also responsible for the incoming data from the data monitoring component, but in that case, the data is also handled by the `SendIBRDTNHandler` that will create a bundle and send it to the IBR-DTN daemon.

3.3.2 Data monitoring

The data monitoring component is receiving different data from the sensors connected to the Raspberry Pi. Those values are directly send to the core application that will write them to the file system and send them to the IBR-DTN daemon. A value is always send with his sensor type, so that the core application knows what kind of value he received.

Since we did not go to the Utah desert this year, we did not had the chance to test it with real sensors. This component is thus generating random data to simulate sensors.

3.3.3 User interface

The user interface will mostly be used at the base station to monitor every astronaut on the field. Figure 3.6 shows an example of the user interface. Here we only have one astronaut (named maxime) and only one data (data.log). Each data type has his own graph on the page

[5]https://github.com/destmaxi/thesis-java-application

of the astronaut. On Figure 3.6 the generated data are random values and do not mean something.

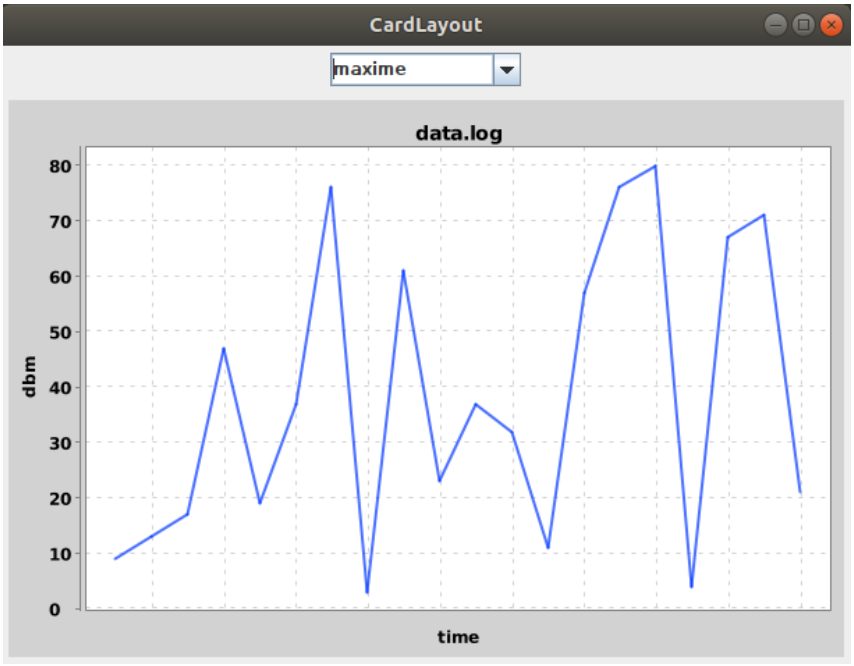


Figure 3.6: User interface

If an astronaut is equipped with a screen, he will be able to see his data and the data that he receives from other astronauts.

Chapter 4

Improving Prophet to aid extra-planetary expeditions

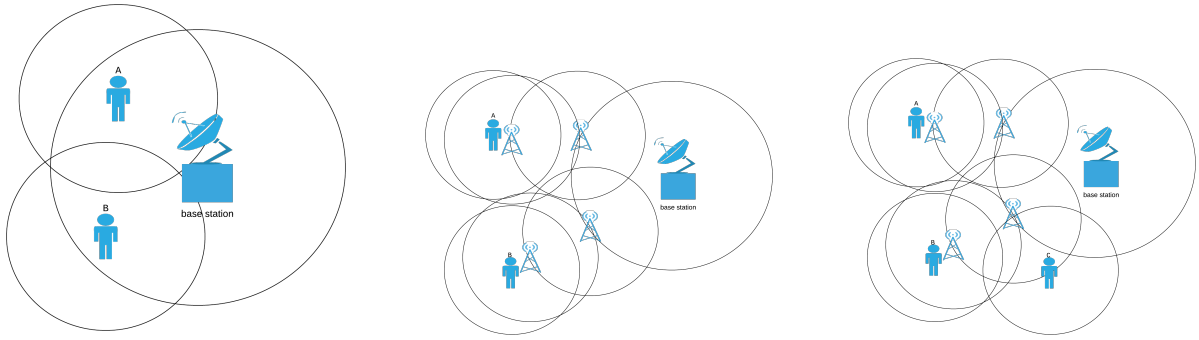
The main problem to aid extraterrestrial explorations is the lack of network coverage due to the range of individual devices and the natural hazards. The key idea of this thesis is to build a static infrastructure to increase the range of individual devices and to maintain a multi-hop path between the astronauts and the base station. There is also a great need in short delays in data transmission. This section will cover in detail how those two needs are satisfied. We will first go through the Backbone placement service, which aim at helping astronauts to place new infrastructure nodes in order to stay in range with the base station. Finally, we will explain the Improved PRoPHET protocol, which aims at reducing the delay for data messages by making use of the static infrastructure.

4.1 Backbone placement problem

In the context of extra-planetary expeditions, astronauts will have to perform Soil Exploration (SE) activities. During these SEs it is crucial to maintain an end-to-end communication with the base station because they may not always take place nearto the base station. A first constraint is that we are limited in coverage by the range of individual wireless devices. In order to increase this coverage, a temporary static infrastructure of fixed devices could be build. But how should it be build? Another constraint of this use case is that we cannot ask an astronaut to change his path, it is defined before the SE and it should not change. Therefore, the infrastructure has to be build based on the route astronauts follow. Another constrain is mobility, each astronaut is constantly moving which makes it even more complicated to maintain a communication between every astronaut and the base station. Indeed, if an astronaut is moving away from other astronauts and from the static infrastructure he should know when to place another fixed device to maintain a path with the base station.

In this context, the *backbone placement problem* refers to the fact of knowing when to place a temporary fixed infrastructure node.

Figure 4.1 shows an example of the backbone placement problem. In Figure 4.1a two astronauts are leaving the base station. When they arrive on the edge of the communication range they need to place a fixed node to increase the coverage with the base station. Without such node they will be out of range and lose communication with the base station. In Figure 4.1b the astronauts place fixed nodes every time they arrived at the edge of the communication range. They have build a fixed infrastructure along their path. Finally, Figure 4.1c shows that two other astronauts are leaving the base station and can use the already placed fixed infrastructure.



(a) Astronauts leaving the base station area

(b) Astronauts placing infrastructure nodes

(c) Other astronauts leaving the station and using the infrastructure

Figure 4.1: Backbone placement problem

4.2 Backbone service

The solution proposed to the backbone placement problem is to monitor the position of the Ad-Hoc nodes with respect to the different infrastructure nodes. If an Ad-Hoc node is getting to far from every infrastructure node, it needs to notify the astronaut that he or she should place a new infrastructure node.

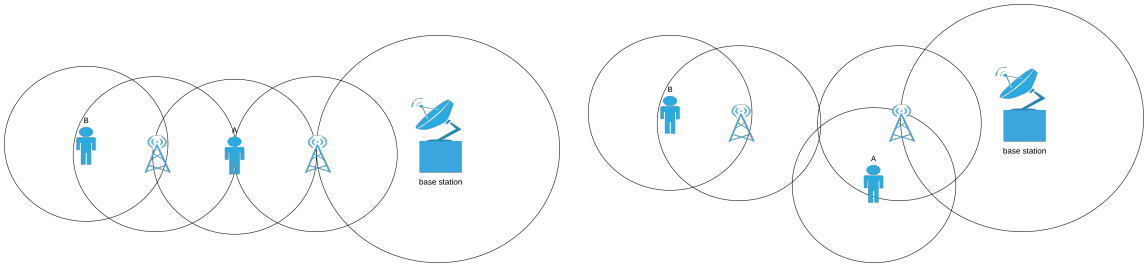
The first contribution of the present work is the design and implementation of the backbone service in order to give an answer to the following questions:

- Is there a more efficient technique that does not require every node to broadcast HELLO messages periodically?
- Is that technique effectively doing it without exhausting the battery of the astronauts?
- And finally, is it doing it without asking the astronauts to go back to position the infrastructure node?

One way to solve the backbone placement problem is that every node (mobile and static) sends periodically a control message, which is commonly known as HELLO message. If a node does not receive a HELLO message for some time, then the node notifies the astronaut that he is out of range and that he should go back to put an infrastructure node.

Another goal of the backbone service is to avoid astronauts to turn back each time they have to place a fixed node. Indeed, the network should be able to detect that a fixed node should be placed before being out of range. Furthermore, having all nodes sending HELLO messages (as explained in the first solution) could lead to the scenario described on Figure 4.2, having a mobile node between two infrastructure nodes, and thus acting as an infrastructure node could lead to a broken end-to-end communication with the base station if the astronaut ever moves. To avoid such situation, only infrastructure nodes should broadcast HELLO messages. This has the following advantages: fewer messages in the network, reduction in energy consumption of the mobile nodes and preventing scenarios where a mobile node is taken into account as an infrastructure node to ensure the end-to-end communication.

It is evident, however, that if infrastructure nodes become isolated, the protocol will no longer ensure an end-to-end communication with the base station.



(a) path between astronaut B and base station (b) broken path due to mobility

Figure 4.2: Broken path due to mobile node acting as infrastructure node

In light of the above, one of the characteristics of the problem is that the astronauts are coming back taking the same path. In addition to this, astronauts will have to go back at the same time to avoid having another group of them removing infrastructure nodes that are mandatory for the end-to-end communication to the base station. The backbone service is an approach to ensure an end-to-end reliable communication between mobile nodes and the base station.

4.2.1 The backbone protocol

In simple terms, the protocol works as follows: infrastructure nodes are broadcasting periodically (every Δ_{hello} time) HELLO messages. Since the assumption that nodes are capable to determine the distance at which a given message was send, was validated in Section 3.1.1 we will of course use it. In addition to this, the communication range of an Ad-Hoc node is known. With that, a node is fully capable to estimate in what time it will be out of range. Indeed, on reception of an HELLO message, mobile nodes are calculating the distance between the infrastructure node and themselves. Based on the distance, they can calculate their speed. With the speed information they are able to estimate in what time they will be out of range of any infrastructure node.

The distance and time at which a mobile node received an HELLO message are stored in a map where the keys are the host ids. Another map is used to store the time before the node will actually be out of range. With this information, a node is capable, at any time, to calculate when he will be out of range of any infrastructure node.

Figure 4.3 shows the graphical approach to predict when a node will be out of range. The slope of a function of distance versus time gives the speed, as Equation 4.1 shows it. Assuming the astronauts are moving at constant speed between two HELLO messages, and knowing the maximum communication range of the infrastructure nodes, it is easy to extrapolate the distance/time function to find the intersection with the maximum communication range. The projection of this intersection will give the time at which the node will get out of range with the given infrastructure node. Equation 4.2 gives the time remaining before getting out of range.

$$\nu = \frac{\Delta x_1}{\Delta t_1} \tag{4.1}$$

$$\Delta t_2 = \frac{\Delta x_2}{\nu} \tag{4.2}$$

Furthermore, the protocol should notify the astronaut that he will get out of range and that he should place a new infrastructure node. In order to know when astronauts might be out of range, it should periodically verify that Equation 4.3 holds. Once Equation 4.3 is satisfied, a notification should be send to the astronaut. For a known velocity ν and the time between

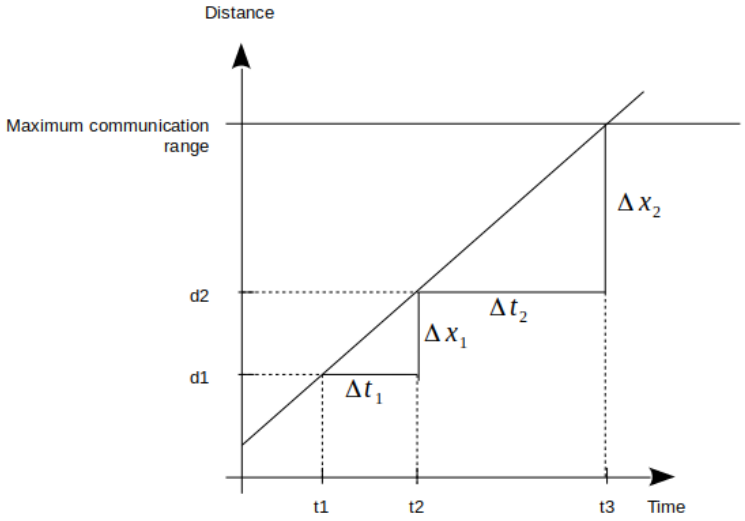


Figure 4.3: Graphical approach of out of range prediction

the reception of HELLO messages Δt_2, ν is positive and Δt_2 is greater than some predefined threshold ($r_{\max}$), then we are getting out of range of all infrastructure nodes. A positive speed means that we are moving away from an infrastructure node.

$$\forall(\nu_i, \Delta t_{2_i})((\nu_i > 0) \wedge (\Delta t_{2_i} > r_{\max})) \tag{4.3}$$

The predefined threshold ($r_{\max}$) depends on the motion of nodes that might let astronauts lose contact with their closest fixed node. A higher value of $r_{\max}$ will result in having several fixed nodes which would have an impact on the delay of routing data packets, while a lower value for it will result in getting out of range before getting the notification to set a new infrastructure node.

Finally, before notifying an astronaut that he should put a new infrastructure node, the Ad-Hoc node must be sure that there is no second infrastructure node around the astronaut. Indeed, if a group of astronauts is moving together and getting out of range at the same time, only one astronaut in the group should place a temporary infrastructure. To do so, before notifying the astronaut the backbone protocol notifies the infrastructure node the identity of such astronaut. The infrastructure node will then decline all requests coming from the same direction for t_{request} time.

Algorithm 2 is our implementation of the backbone service running on mobile nodes. Each time a mobile node receives a HELLO (line 10) message from an infrastructure node, it updates its distance (line 12) map and the time (line 13) at which it received the message. As explained above, those messages are necessary for Ad-Hoc nodes to keep track of their distance with infrastructure nodes. On the reception of every two messages (line 14), the time before the node will get out of range will be calculated according to Equation 4.2 (lines 15-17) and an update to the corresponding map will take place (line 18). We decided to evaluate only every two messages since each time a HELLO message is received a timer is set and we do not want to have to much timers created at the same time. Finally, a new timer (line 19) is created and expires after Δ_{check} time. Once the timer has expired, the node will check if it satisfies Equation 4.3 (line 24). If it does, it means that the mobile node is getting out of range and that it should place a new infrastructure node. Notice that nodes can not decide on their own whether they should place

a new infrastructure node since they lack of information about other mobile nodes. Therefore, once an infrastructure node satisfies Equation 4.3, the node will send a permission request (Perm packet) to the infrastructure (line 25), which is positioned at the shortest relative distance, to satisfy Equation 4.3. If the response is positive, the astronaut can place a new infrastructure node.

Algorithm 2 Backbone service on ad hoc node

```
1: constants
2:     Tx : double                                              ▷ node transmission range
3: variables
4:     r_max : double                                           ▷ out-of-range threshold
5:     Δ_check : double                                         ▷ period to start out-of-range procedure
6:     c_{n_i} ← 0 : integer                                    ▷ counts receptions of hello messages from infrastructure node n_i
7:     FarOffLen : map [n_{i≠j}] → double                       ▷ latest distance to n_i
8:     LastHelloMsg : map [n_{i≠j}] → double                    ▷ latest reception of Hello message from n_i
9:     LastOoRhistory : map [n_{i≠j}] → double                  ▷ lastest out-of-range approximation of n_i
10: procedure MAC.receive(Hello message h from n_i)
11:     c_{n_i} ← c_{n_i} + 1
12:     d ← aproxDistanceTo(n_i)                                 ▷ approximates relative distance to infrastructure node
13:     ts ← getTimestamp(h)                                     ▷ gets time stamp of reception
14:     if c_{n_i} mod 2 = 0 then
15:         Δ_distance ← abs(d - FarOffLen[n_i])
16:         Δ_time ← abs(ts - LastHelloMsg[n_i])
17:         v ← Δ_distance / Δ_time
18:         LastOoRhistory[n_i] ← Tx / v
19:         t_{n_i} ← new timer with duration Δ_check
20:     else                                                     ▷ the first time we can't decide when we will be out of range
21:         FarOffLen[n_i] ← d
22:         LastHelloMsg[n_i] ← ts
23: when timer t_{n_i} expires
24:     if LastOoRhistory[n_i] <= r_max and aproxSpeed(n_i) > 0 then
        // chose the most critical host
25:         select n_j ∈ keys(FarOffLen) s. t. FarOffLen[n_j] is the maximum distance in FarOffLen
26:         send(new Perm, n_j)
```

An infrastructure node has a list of relative positions where new infrastructure nodes were placed. When this node receives a Perm packet it verifies that it did not already give the permission to set a new node in that area. If it is the case, then it declines the permission and allows the mobile node to set a new infrastructure node. The area is defined by the range of the devices. It is worth to notice that the backbone protocol does not require the propagation of messages through hosts.

4.3 Improving PRoPHET for soil explorations

The routing protocol PRoPHET is commonly used in DTN architectures, as described in Section 2.3.1, this protocol is based on the assumption that, if a node B have been within the transmission range of a second node A, it is likely that A will continue to see B in the future. In soil explorations the proposed temporary fixed infrastructure is build on demand and the use of PRoPHET will result in observing constant values of high predictability for routing in fixed nodes. It is thus important to change such behaviour in order to give priority to mobile nodes.

4.3.1 Proposed improvement in PRoPHET

The second contribution in this research work is an adaptation to the PRoPHET routing protocol. The main idea is to take advantage of the fixed nodes in order to quickly route messages to their destinations. This is an adaptation to the routing protocol to aid soil explorations in order to

reduce latency of messages. If something goes wrong with an astronaut, the base station should know it as soon as possible to be able to act accordingly.

Section 2.3.1 shows how the delivery predictability is calculated and one can notice that there are different constants that are involved in the calculation. Changing those constants changes the evolution of the predictability. Since the GTMX forwarding strategy chooses to forward messages to nodes with a high predictability, increasing the probability of a node will increase the probability that messages are routed towards him.

The predictability depends on three factors, the initial value given when two nodes encounter for the first time ($P_{\text{first\_encounter}}$), the number of times two nodes encounter ($P_{\text{encounter\_max}}$ defines the number of times two nodes have to encounter to reach a certain level) and finally PRoPHET benefits of the transitive property. The β parameter adjusts the weight of the transitive property of PRoPHET, that is, how much impact we gave to information about destinations that are received from encountered nodes. If β is set to zero, then the transitive property will not be activated and only direct encounters will be taken into account in the calculation of the delivery predictability. To increase the importance of information learned by fixed nodes, in the context of soil explorations, it is required to increase such value only for infrastructure nodes. A mobile node would then have two ways to calculate the delivery predictability, one for mobile nodes (with a smaller β) and one for static nodes (with a greater β).

It is also required to set a value for the first encounter ($P_{\text{first\_encounter}}$. PRoPHET does not, by default, make any assumptions about the frequency at which nodes will be repeatedly in contact. When two nodes encounter, and it is the first time they have seen each other or the first time in a while, they set the delivery predictability of the encountered node to $P_{\text{first\_encounter}}$. Previous studies suggests [1] to use 0.5 if there is no extra information about node encounters. In the context of soil explorations, when two infrastructure nodes encounter for the first time (when they are placed), they will remain in the same position. This let us modify $P_{\text{first\_encounter}}$ and even set it to one since we are sure they will stay neighbours.

Changing only the β and $P_{\text{first\_encounter}}$ parameters will not really increase the routing time. It will only help the PRoPHET routing protocol to select a route through the static infrastructure. The improved PRoPHET routing protocol combines the optimisation of the β and $P_{\text{first\_encounter}}$ parameters with the idea of directly forwarding message through infrastructure nodes. The next section will go more into detail on the implementation and value selection.

Implementation

As explained above, the improved PRoPHET routing protocol modifies two aspects of the initial protocol. On one hand, it modifies the way the delivery predictability map is updated when the encountered node is a static node and on the other hand it changes the way messages are forwarded between static nodes.

Algorithm 3 is our implementation of the proposed improvement in PRoPHET for soil explorations. When a node receives a response to a handshake request (line 13) it will update his local delivery predictability map (dpm) as explained in Section 2.3.1 (line 19). Only, the β and $P_{\text{first\_encounter}}$ (PEF) take different values depending on the kind of nodes that are encountered. Lines 4-8 show the different parameters, β_{imp} and PEF_{imp} are the modified values. As explained, depending on which type of node is receiving the message (Ad-Hoc or infrastructure node) and from what type of node it received the message, either β (PEF) will be used or β_{imp} (PEF_{imp}). Therefore both Ad-Hoc nodes and infrastructure nodes have a different implementation of the `updateDpm` procedure. Line 20 shows the procedure followed by mobile nodes, in this case we only

use PEF because even if the node encounters an infrastructure node, nodes are mobile and might be out of range at any time. On the other hand, if the encountered node is an infrastructure node (line 25) the β_{imp} parameter is used (line 26). Finally, line 29 shows the procedure followed on infrastructure nodes. In this case, the value PEF_{imp} will be used when another infrastructure node will be encountered since it will never move (line 30-33); more details on those parameters are given below. On a handshake request the node will also go through all its stored messages and verify that it does not have a message to forward to the just encountered node (lines 15-16). When a message need to be forwarded, it is calculated as explained in Section 2.3.1.

Algorithm 3 Improved PRoPHET protocol

```
 1: constants
 2:     nodeId: int                                                          ▷ local node id
 3:     δ: double          ▷ sets the maximum value of the delivery predictability for a destination
 4:     PEF: double          ▷ delivery predictability when two nodes encounter for the first time
 5:     PEF_imp: double                                  ▷ PEF between two infrastructure nodes
 6:     β: double                                ▷ adjust the weight of the transitive property
 7:     β_imp: double                ▷ β for interactions between mobile and infrastructure nodes
 8: variables
 9:     age: map, init empty                ▷ key: nodeId, value: time at which they last encounter
10:     messages: vector, init empty                    ▷ list of messages that need to be forwarded
11:     dpm: DeliveryPredictabilityMap, init empty      ▷ key: host id, value: delivery predictability
12:     strategy: ForwardingStrategy, init GTMX      ▷ the forwarding strategy used to forward messages
13: procedure MAC.receive(Packet p from n_i)                  ▷ response to handshake packet
14:     srcDpm ← p.getDpm()
15:     ∀ m_j ∈ messages where strategy.shallForward(srcDpm, m_j, n_i)
16:     send(m_j, n_i)
17:     dpm.age()                                          ▷ age the local delivery predictabilities
18:     age[n_i] ← getTimestamp(p)
19:     updateDpm(n_i, srcDpm)                          ▷ update the local delivery predictabilities
```

Mobile node

```
20: procedure updateDpm(n_i, srcDpm)
21:     P_ab ← (1 - δ - P_ab_old) * p_encounter(n_i)
22:     if firstEncounter(n_i) then
23:         P_ab ← PEF
24:     dpm[n_i] ← P_ab
25:     if isInfrastructureNode(n_i) then
26:         dpm.update(β_imp, PEF, srcDpm, n_i)
27:     else
28:         dpm.update(β, PEF, srcDpm, n_i)
```

Infrastructure node

```
29: procedure updateDpm(n_i, srcDpm)
30:     if isInfrastructureNode(n_i) then
31:         EF ← PEF_imp
32:     else
33:         EF ← PEF
34:     P_ab ← (1 - δ - P_ab_old) * p_encounter(n_i)
35:     if firstEncounter(n_i) then
36:         P_ab ← EF
37:     dpm[n_i] ← P_ab
38:     if isInfrastructureNode(n_i) then
39:         dpm.update(β_imp, PEF_imp, srcDpm, n_i)
40:     else
41:         dpm.update(β, PEF, srcDpm, n_i)
```

As suggested in [1], we use the following values:

$$\beta = 0.9$$
$$PEF = 0.5$$

The value of β is already very high since it would not make sense to use the PRoPHET routing protocol without its transitive functionality. Indeed, PRoPHET is powerful because nodes not only learn about the network from what they see but because they also learn from what other nodes have seen.

In the IMPROVED UPDATE method (update method used in our implementation) for mobile and infrastructure nodes, the parameters have the following values:

$$\beta_{\text{imp}} = 1$$
$$\text{PEF}_{\text{imp}} = 1$$

The PEF_{imp} parameter is not used on mobile nodes since there is no information about the motion of mobile nodes. When two infrastructure nodes encounter, they will stay in range and therefore they can set the PEF_{imp} parameter to one. On the other hand, the β_{imp} parameter is used on both mobile and infrastructure nodes. Indeed, in both cases, when they encounter an infrastructure node, they should use the improved β value. Setting a high value for β (β_{imp}) when encountering an infrastructure node will favour routing towards the infrastructure node.

As explained before, the improved PRoPHET routing protocol differs from the classical one by two things. First by the modification of β and $P_{first\_encounter}$. when an infrastructure node receives a message, it will directly ask the forwarding strategy to which node the message should be forwarded. If that node is an infrastructure node then the message is stored and directly forwarded to him, otherwise the message is just stored and will be forwarded on an encounter with a mobile node.

The proposed adaptation of PRoPHET takes advantage of the static infrastructure by favour routing message through it and by not waiting for an encounter to happen to forward messages.

Chapter 5

Evaluations

In this chapter we will focus on the evaluation of the performances in terms of latency and throughput of the monitoring service and improved prophet protocol. The simulations are using a simple MAC protocol and they mimic outside activity use cases. This implies that nodes are moving like a typical outside activity would go. Typical outside activities are explained in more details in Section 1.1. The rest of the chapter is organised as follow, first we will cover some technical details about the simulator and the simulations, then we will have a mini-benchmark of the original PRoPHET (latency and throughput) and then with the improved PRoPHET to show the impact of it. Finally we will evaluate different outside activity use cases.

5.1 Technical details of experiment, simulator, ...

The evaluations were made with the omnet++[14] network simulator and the INET framework[15]. All the simulations use the *AckingWirelessInterface*[1] from the INET framework. It is a simple wireless NIC that consists of a unit disk radio and a trivial MAC protocol. We chose this wireless interface since it offers simplicity for scenarios where Layer 1 and 2 can be completely ignored, which is the case for this thesis. Furthermore, it allows us to simply modify the communication range of each device. The hardware used has only one transceiver and one emitter. A node can not receive two messages at the same time neither send two (or more) messages at the same time. This implies that messages and control packets need to be queued before being send.

First of all we will define what *throughput* and *latency* means for this chapter. The throughput is the number of messages that a node can send in one second. By messages we understand only data packets, not the control packets. The latency, however, is the time that the network needs to route a message from the source node to the destination node. That time is measured once the message is send, thus after the queue until the message is received by the destination node.

The queuing of control packets and messages lead us to the first parameter that has a big impact on the latency and throughput. Δt_{send} is the time interval at which messages in the queue are send. Even if this parameter has a big impact on the results, we will not cover it in details since it has nothing to do with the solutions proposed in this thesis. For the rest of this chapter we will consider that $\Delta t_{\text{send}} = 10[ms]$.

Both the backbone service and the improved PRoPHET protocol have some parameters that have also a critical impact on the latency and throughput. The backbone service sends every Δt_{hello} time interval a Hello packet. As stated above, both messages and control packets are in the same queue, if this queue is full of Hello packets, it will have an impact on the throughput of

[1] https://doc.omnetpp.org/inet/api-current/neddoc/inet.linklayer.acking.AckingWirelessInterface.html

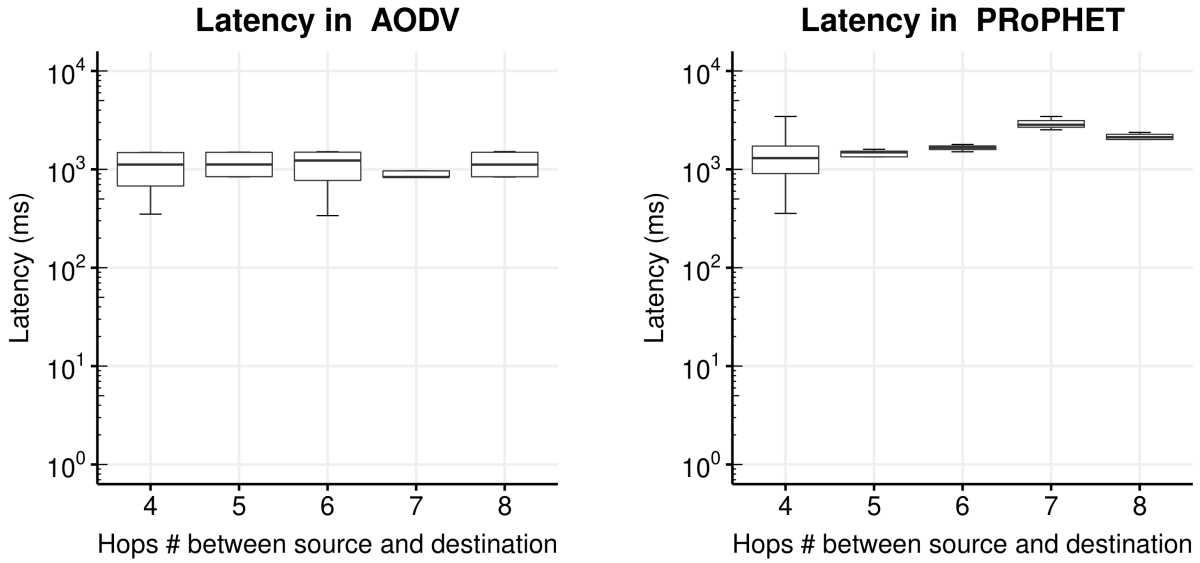
messages. On the other hand, having a to big Δt_{hello} will lead to a malfunction of the backbone protocol by having astronauts getting out of range and not detecting it. The improved PRoPHET routing protocol also sends every Δt_{rhand} time interval a request handshake packet. Those have not only an impact on the throughput just as the Hello packets but also on the latency of messages. Indeed, each time a node receives a request handshake packet they will search for messages to forward to the encountered node. Having a small Δt_{rhand} will improve the latency of routing messages but will also decrease the throughput. On the other hand, having a big Δt_{rhand} will lead to a better throughput but also a bigger latency.

5.2 Mini benchmark of PRoPHET

In this section we will evaluate our implementation of the classical PRoPHET routing protocol. Indeed, the implementation of the classical PRoPHET routing protocol was not yet implemented in the omnet++ network simulator. Therefor we had to first implement it before improving it.

To evaluate and validate our implementation we compared it with a well known routing protocol in MANETs, the AODV routing protocol. As explained in Section 2.3, the AODV routing protocol is a reactive routing protocol. Before sending a data packet the source node will first find a route in the network by spreading a route request packet (RREQ). Nodes other than the destination receiving a RREQ extend the content of this packet with their identifier. When the destination finally receives the RREQ request, it replies to the source with a route reply packet (RREP).

The scenario used for the evaluation is the following; our deployment contained 100 nodes and originally every node is located at a random position in a 100mx100m area. Nodes mimic human movement by performing truncated levy walks, that is, which combines frequent short walks with occasional rides to distant locations. The velocity of every walk is chosen randomly from an interval: [0:1] m/s. Every node has a transmission range of 10m. For the benchmark we took every time 10 routes where the hops distance between source and destination varies between 4 and 8 hops.



(a) Latency of AODV routing protocol

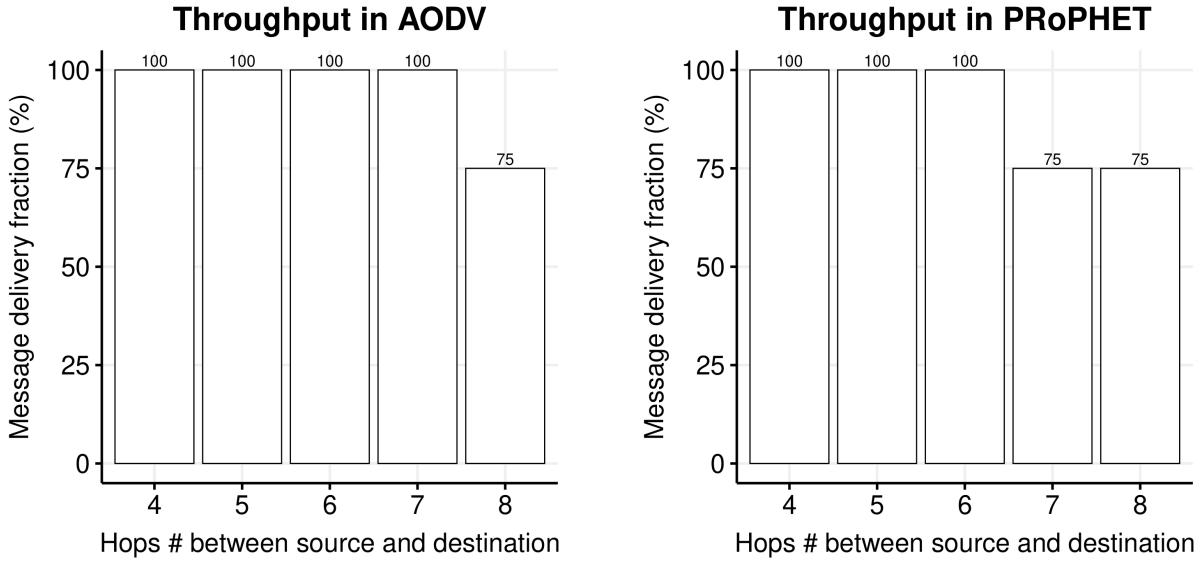
(b) Lantecy of PRoPHET routing protocol

Figure 5.1: Latency of AODV and PRoPHET routing protocol

To evaluate our implementation of the PRoPHET routing protocol we measured his throughput and latency. The throughput is the number of data messages received by the destination. The latency is the time elapsed from the moment where the data message leaves the source node to the moment it arrives at the destination node. Figure 5.1 shows the results for both PRoPHET and AODV.

As we can see in Figure 5.1, the average time for a message to arrive at the destination is quite the same for PRoPHET and AODV. Figure 5.1a shows the results of the simulation for the AODV routing protocol. We can observe that the average time to send a message does not really depend on the number of hops between the source node and the destination node. This can be explained by the fact that once the route is discovered, the message is directly forwarded, hop-by-hop up to the destination. Finding the route is most of the time also quite fast but could take some time if the network is partitioned at the moment of sending the message. Indeed, the AODV routing protocol must have a route between the source and destination node at the time of sending the message. The variance for AODV is thus quite important since we are in a mobile network and the network may be partitioned for a short time.

For the PRoPHET routing protocol it is quite different. Although the average time is mostly the same as for PRoPHET, it does increase with the number of hops between the source node and the destination node as we can see on Figure 5.1b. This can be explained by the fact that in the PRoPHET routing protocol, messages are only forwarded when two nodes encounter. Thus how greater the number of hops between the source and destination node is, how greater the latency will be. On the other hand, the variance is very low for PRoPHET, this is because it does not care about partitioned networks, a node will carry the message until it encounters a node that has a better probability than it to encounter the destination node. As we can see on Figure 5.1b, the variance when we have 4 hops is quite big. There is no good reason for this, a possible explanation is that the network must have been partitioned a lot and that the source node (or any other node between the source node and destination node) did not encounter other nodes that had already seen the destination node.



(a) Throughput of AODV routing protocol

(b) Throughput of PRoPHET routing protocol

Figure 5.2: Throughput of AODV and PRoPHET routing protocol

Figure 5.2 compares the throughput of both AODV and PRoPHET. Here we have measured

the message delivery fraction in function of the number of hops. We can see that both AODV and PRoPHET have a good delivery fraction. As Figure 5.2a and 5.2b show, the throughput is of 100% for the three number of hops. Since the distance is not that great, the probability to have message losses or collision is quite small. AODV is loosing in performances as of 8 hops while for PRoPHET, it begins to loose in performances from 7 hops. This can be explained by the fact that in the configuration of PRoPHET, every message is only send once per encounter node. When there is a great number of hops between the source and destination the probability that a node could not forward his message duo to a collision increases. For AODV this is not the case since it uses flooding to find a route and once the route is found messages are send directly to the destination. The reason why in AODV we could have message losses is if the sender finds a route and that after the message was send, the route breaks due to mobility. Here again, the probability that this occurs increases with the number of hops.

We can finally say that our implementation of the PRoPHET routing protocol validated is. Indeed, the latency of messages is what is expected of a routing protocol in mobile Ad-Hoc networks and the throughput is also quite similar to AODV. Now that we have validated our implementation, we can compare it with our improved PRoPHET version.

5.3 Use case description

In this section we will evaluate the contribution in scenarios that describe a real use case. First we will compare the classic PRoPHET routing protocol with our improved version. The improved version should have a lower latency for messages than the classical PRoPHET since it takes advantage of the fixed infrastructure. Finally, we will evaluate our backbone placement service.

5.3.1 Improved PRoPHET routing protocol

To evaluate the improved PRoPHET routing protocol we took a scenario where an astronaut (Ad-Hoc node) was between 4 and 8 hops from the base station, in between the base station and the astronaut there were thus between 4 and 8 fixed infrastructure nodes depending on the scenario. The objective of this evaluation is to see if taking the infrastructure into account is indeed increasing the performances. The simulation was done with a range of 100 [m] for each node.

Table 5.1 shows the results of the simulations for both PRoPHET and Improved PRoPHET. As we can see, our Improved PRoPHET protocol is a lot faster than our classic PRoPHET implementation. This can be explained by the fact that the improved PRoPHET routing protocol does not wait for an encounter to happen when a message needs to be forwarded between two infrastructure nodes. Once the message is in the infrastructure it is directly routed to the destination, in this case the base station.

Number of hops	Latency PRoPHET [s]	Latency improved PRoPHET [s]
4	1.602	0.32
5	1.4	0.34
6	1.503	0. 33
7	2.937	0.36
8	2.16	0.41

Table 5.1: Latency for PRoPHET and Improved PRoPHET in [s]

Those results show that our implementation indeed offers a smaller latency in message delivery. This is a crucial aspect of the Mars use case and is one of the main contributions of this thesis.

5.3.2 backbone service evaluation

Evaluating the backbone service is quite difficult. To properly evaluate it, nodes should be able to dynamically create new nodes in the simulator. We had not enough time to implement this feature in the simulator and we thus had to make some small changes in the service to be able to evaluate it. An infrastructure nodes should be activated to broadcast Hello packets. When an Ad-Hoc node has to put a new infrastructure node it will broadcast an Activate packet and the infrastructure node will then be activated and begin to broadcast Hello packets. That way, we artificially created a new node in the network. Furthermore, to be able to actually evaluate if the backbone protocol precise is (the new node should be created as close as possible to the max range of the previous infrastructure node) we logged the position at which the Ad-Hoc node was when broadcasting the Activate packet.

$$2 * \Delta t_{\text{hello}} <= r_{\max} \tag{5.1}$$

The backbone service has two parameters that influence its results. One is the Δt_{hello}, the time interval at which Hello packets are send and the other is $r_{\max}$, the maximum time before being out of range. For the backbone protocol to work correctly they must respect Equation 5.1. This is mandatory because otherwise an Ad-Hoc node may not receive a Hello message before being out of range. It is 2 * Δt_{hello} since we only evaluate if we are out of range every two Hello packet (see Algorithm 2).

Figure 5.3 shows the use case we used to evaluate the backbone service. There is one base station (host[0]) that is activated, 5 infrastructure nodes (host[1-5]) that are initially deactivated and one Ad-Hoc node (adhoc[0]). Each infrastructure node is positioned at the edge of the range of the previous infrastructure node. This is the optimal distance where to place a new node since it will optimise the global coverage of the network. The Ad-Hoc node starts at the base station and move with a constant speed of 1 [m/s], which is slightly less then the average human walking speed (1.42 [m/s]) [16]. We chose a speed slightly less since astronauts have heavy suits which will slow them down. The Ad-Hoc node moves in straight line between nodes. The total simulation takes 500 seconds, which is the time the Ad-Hoc node need to reach the last infrastructure node. The values for the two parameters (Δt_{hello} and $r_{\max}$) where the following.

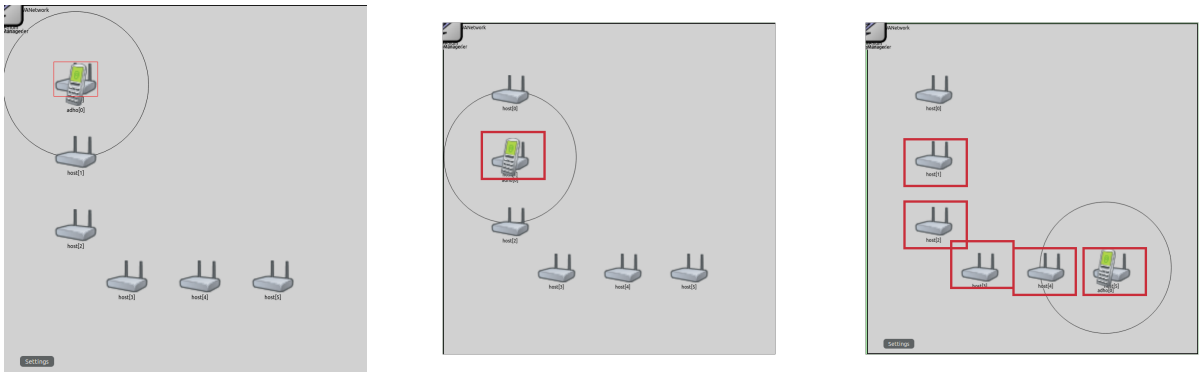
$$\Delta t_{\text{hello}} = 2$$
$$r_{\max} = 5$$

node 1	(100, 196.53)
node 2	(100, 296.25)
node 3	(167.86, 367.86)
node 4	(265.40, 370)
node 5	(365.48, 370)

Table 5.2: New infrastructure node positions (x, y)

The initial stage is shown on Figure 5.3a, the Ad-Hoc node is at the base station and only the base station is broadcasting Hello packets. The Ad-Hoc node will start moving towards the first infrastructure node (host[1]), each time it receives an Hello packet it verifies that it is not out of range as explained in Section 4.1 with Algorithm 2. On Figure 5.3b the Ad-Hoc node

reaches the critical position, it is almost out of range with the base station. As shown on the figure, the Ad-Hoc node detects that and broadcast an Activate packet which activates the first infrastructure node (host[1]). Table 5.2 contains the different positions where the Ad-Hoc node has placed new infrastructure nodes. The first node was placed on position (100, 196.53) instead of (100, 200), that is the optimal position. Once the first node is activated it starts broadcasting Hello packets on its own and the Ad-Hoc node goes further. Figure 5.3c shows a position where the Ad-Hoc node reaches the end of the simulation and has activated all infrastructure nodes.



(a) (b) (c)

Figure 5.3: Backbone service evaluation

If we take a closer look to Table 5.2 we can see that the backbone service quite precise is. The expected values are shown in Table 5.3. The backbone protocol places new infrastructure nodes +- 5m before the limit. We could maybe have better results by tuning the parameters a bit more but we should not forget that in a real use case the astronaut is confronted to natural hazards and may need to set the node a bit further.

node 1	(100, 200)
node 2	(100, 300)
node 3	(170, 370)
node 4	(270, 370)
node 5	(370, 370)

Table 5.3: Ideal new infrastructure node positions (x, y)

Table 5.4 show the node positions when Equation 5.1 is not respected, the values chosen for those results where the following.

$$\Delta t_{\text{hello}} = 3$$
$$r_{\max} = 5$$

As we can see in Table 5.4, node 3 was never set by the Ad-Hoc node and by consequence node 4 and 5 neither. Node three was not set because the last Hello packet that the Ad-Hoc node received from node 2 (host[2]) was a Hello packet that was not taken into account (recall Algorithm 2). The next Hello packet that it should have received would have been 3 seconds later but since the max time out of range ($r_{\max}$) is only 5 seconds. Since the Ad-Hoc node did not took into account the last Hello packet it will already be out of range when it receives the next one and thus never trigger the Activate packet. One could ask why not take into account

node 1	(100, 196.53)
node 2	(100, 296.25)
node 3	????
node 4	????
node 5	????

Table 5.4: New infrastructure node positions (x, y)

every Hello packet. Unfortunately this will not solve the problem, Equation 5.1 would just need a small adaptation and the problem will remain. Equation 5.2 shows the equation in case we would take every Hello packet into account.

$$\Delta t_{\text{hello}} <= r_{\max} \tag{5.2}$$

After having evaluated and discussed the impact of both the Δt_{hello} and the $t_{\max}$ parameters we can say that how smaller they are, how preciser the backbone service will be. Nevertheless, we should not forget that the goal is to simulate real case scenarios and it is a good practice to give some error margin. On a simulator we could tune the parameters so that the backbone service would place the infrastructure nodes on the best place since everything can be 100% predicted. On the field, we should take into account the unpredictability and therefore give some error margin to the service to be sure the astronaut would not get out of range before setting a new node.

Chapter 6

Conclusion

In this thesis we presented a solution to aid extra-planetary expeditions where astronauts communicate with the base station. In such expeditions there is no available communication infrastructure and astronauts are thus limited by the range of individual devices. To maintain reliable routes to the base station, astronauts build a temporary infrastructure of fixed nodes. Given that astronauts are constantly moving and may even create partitioned networks, the proposed hybrid approach is based on the Delay Tolerant Network (DTN) architecture.

This thesis contains two main contributions, first the backbone placement protocol that monitors the position of the astronaut according to the already positioned infrastructure nodes. When an astronaut is about to be out of range, the service will detect it and send a notification to the astronaut in order to indicate him to place a new infrastructure node. The second contribution is an improvement to the PRoPHET routing protocol by taking into account the infrastructure nodes and favour routes through this infrastructure.

Before we could evaluate our improved version of PRoPHET we had to implement and validate the classical PRoPHET protocol since it was not present in the network simulator used. We compared our implementation of PRoPHET with a well known MANET routing protocol, Ad-Hoc On Demand Distance Vector (AODV). Our implementation of PRoPHET has, as expected, a bit more latency in average and the same throughput. Once our implementation was validated, we evaluated our improved version and the backbone placement service.

The improved PRoPHET routing protocol has a much lower latency than the classical PRoPHET protocol, it is up to five times faster. Which validates the protocol. The backbone placement protocol places the infrastructure nodes with an error margin of 5%, as explained in the Evaluation Section, this is a wanted behaviour since on the field you must take into account some unexpected outcomes. Those results are very promising for extra-planetary expeditions.

We validated our approach through simulations and in March 2021 we will be able to test it in the Utah desert at the Mars Desert Research Station (MDRS) with a real deployment.

Bibliography

[1] A. Lindgren, E. Davies, S. Grasic, and A. Doria, "Probabilistic Routing Protocol for Intermittently Connected Networks."

[2] "Exploration of Mars," July 2020. Page Version ID: 970538778.

[3] S.-J. Lee and M. Gerla, "AODV-BR: backup routing in ad hoc networks," in *2000 IEEE Wireless Communications and Networking Conference. Conference Record (Cat. No.00TH8540)*, (Chicago, IL, USA), pp. 1311–1316, IEEE, 2000.

[4] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," *ACM SIGCOMM Computer Communication Review*, vol. 24, pp. 234–244, Oct. 1994.

[5] "Institut national de recherche en informatique et en automatique," *Bulletin of Sociological Methodology/Bulletin de Méthodologie Sociologique*, vol. 37, pp. 55–57, Dec. 1992.

[6] K. Fall, "A delay-tolerant network architecture for challenged internets," in *Proceedings of the 2003 conference on Applications, technologies, architectures, and protocols for computer communications*, SIGCOMM '03, (Karlsruhe, Germany), pp. 27–34, Association for Computing Machinery, Aug. 2003.

[7] J. Kolodziej, S. U. Khan, L. Wang, N. Min-Allah, S. A. Madani, N. Ghani, and H. Li, "An Application of Markov Jump Process Model for Activity-Based Indoor Mobility Prediction in Wireless Networks," in *2011 Frontiers of Information Technology*, pp. 51–56, Dec. 2011.

[8] "IRTF Delay-Tolerant Networking Research Group (DTNRG) [CONCLUDED]."

[9] J. Ott, D. Kutscher, and C. Dwertmann, "Integrating DTN and MANET routing," in *Proceedings of the 2006 SIGCOMM workshop on Challenged networks - CHANTS '06*, (Pisa, Italy), pp. 221–228, ACM Press, 2006.

[10] K. U. R. Khan, R. U. Zaman, A. V. Reddy, K. A. Reddy, and T. S. Harsha, "An Efficient DSDV Routing Protocol for Wireless Mobile Ad Hoc Networks and its Performance Comparison," in *2008 Second UKSIM European Symposium on Computer Modeling and Simulation*, (Liverpool, UK), pp. 506–511, IEEE, Sept. 2008.

[11] S. Jain, K. Fall, and R. Patra, "Routing in a Delay Tolerant Network," p. 13.

[12] S. Schildt, J. Morgenroth, W.-B. Pöttner, and L. Wolf, "IBR-DTN: A lightweight, modular and highly portable Bundle Protocol implementation," *Electronic Communications of the EASST*, vol. 37, Feb. 2011.

[13] K. L. Scott <kscott@mitre.org>, "Bundle Protocol Specification."

[14] "OMNeT++ Discrete Event Simulator."

[15] "INET Framework - INET Framework."

[16] R. C. Browning, E. A. Baker, J. A. Herron, and R. Kram, "Effects of obesity and sex on the energetic cost and preferred speed of walking," *Journal of Applied Physiology*, vol. 100, pp. 390–398, Feb. 2006. Publisher: American Physiological Society.

Rue Archimède, 1 bte L6.11.01, 1348 Louvain-la-Neuve **www.uclouvain.be/epl**