

Ad-hoc network possibilities inside LoRaWAN

DANIEL LUNDELL

MASTER'S THESIS

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY

FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY



Ad-hoc network possibilities inside LoRaWAN

Daniel Lundell
daniellundell191@gmail.com

Department of Electrical and Information Technology
Lund University

Supervisor: Emma Fitzgerald, EIT LTH
Anders Hedberg, Sensefarm AB

Examiner: Christian Nyberg, EIT LTH

June 21, 2017

Abstract

The Internet of Things (IOT) is a fast growing field with new actors constantly joining in. Locations such as farms or remote areas do not always have Internet coverage to access the IoT. This thesis looks at LoRaWAN as an IoT technology and ad-hoc networking to solve this problem. Existing ad-hoc routing protocols such as AODV, HWMP, and ZRP were studied. Based on the study they were evaluated as to how well they would fit into the LoRaWAN protocol. A simple solution based on HWMP and AODV was integrated with LoRaWAN. A testbed consisting of LoRaWAN devices was built to test the capabilities of the proposed solution. Receive windows of 2 seconds can be achieved with an ad-hoc LoRaWAN with a depth of 5-6 nodes. Successful routes dropped from 85% to 40% with a depth increase of 2 nodes. LoRaWANs 1% duty cycle limit can be broken with bigger networks. The thesis concludes that ad-hoc LoRaWAN based on HWMP and AODV might be possible given further research. The networks can be used to cover large remote areas with no Internet connection.

Popular Science Summary

Internet of Things(IoT) is a constantly growing and evolving technology. IoT is the concept of connecting multiple devices and allow them to communicate and exchange information. Different companies are competing for the market and new areas where IoT can be applied are still being discovered. New competing IoT technologies are also constantly entering the IoT arena, for example ZigBee, SigFox and LoRaWAN. They are all technologies trying to provide infrastructure for IoT.

Sensefarm AB in Lund is one of those companies, they work with IoT in agriculture. They are currently developing IoT with the help of LoRaWAN. Working with IoT in agriculture involves some challenges due to the often remote working locations at farms. One of the challenges is reliable internet communication which is a central aspect of IoT. Devices are often used to measure temperature, rain or pollution. A typical application is monitoring different values such as temperature or water level. The application needs to alert someone when a temperature gets to high or low. Thus, it needs to know that it can communicate with its devices to provide good and reliable data.

Today there already exist technologies to deal with unreliable and unstable networks with no previous infrastructure; multi-hop networks. In a multi-hop network, every device can communicate with the use of the other devices. They provide routing for each other so that two devices physically far away from each other can communicate using devices between them. This technique can be used together with LoRaWAN to create IoT networks that do not need the internet to deliver data. More specifically it can be applied to the gateways inside a LoRaWAN network. Gateways are bridges between all the monitoring devices and the application often located on a remote server. The communication with the remote server is the unstable connection that needs to be solved. By applying multi-hop technology to the gateways, they can become smarter and not be just bridges to the internet but also bridges for each other to reach a gateway with internet. Technically they can send data between them without ever knowing about the internet. This can be used to apply IoT infrastructure without the need of the Internet, building networks with incredible range and good reliability in remote areas.

Acknowledgment

This thesis would not have been possible if it were not for a number of dedicated persons. Firstly, I would like to thank Anders Hedberg at Sensefarm for giving me the opportunity doing this thesis, furthermore I would like to thank Johnny and Simon, also at Sensefarm for their help. I would also like to thank my adviser at LTH, Emma Fitzgerald for supervising this thesis and answering all my questions.

Finally, I would like to express my sincere gratitude to my mother for always being supportive throughout my education.

Table of Contents

1	Introduction	1
1.1	Background	1
1.2	Limitations	2
1.3	Method	2
1.4	A brief word about taxonomy	3
1.5	Related work	3
1.6	Thesis outline	4
2	Network technologies	5
2.1	Wireless ad-hoc networks	5
2.2	Introduction to the OSI	5
2.3	Standards and amendments	6
2.4	Wireless multi-hop network structures	6
2.5	Routing in ad-hoc networks	8
2.6	LoRa/LoRaWAN	14
2.7	Implementation device	20
3	Prototype/Model	21
3.1	Considerations	21
3.2	Suitable wireless multi-hop technology	23
3.3	Proposed model/protocol	23
4	Experiment, testing and evaluation	31
4.1	Testing	31
4.2	Experiment	34
5	Discussion	39
5.1	Timing	39
5.2	Data overhead	40
5.3	Successful routes	40
5.4	Experiment	40
5.5	Integrating directly into LoRaWAN protocol	41
6	Future work and conclusion	43

6.1	Routing	43
6.2	Security	45
6.3	Conclusion	45
References _____		47
A Test setup _____		51
A.1	Graphical overview	51
A.2	LoPy details	51

List of Figures

1.1	LoRaWAN compared to other network solutions (Source: [1])	1
2.1	OSI model (Source: IBM Knowledge Center)	6
2.2	Different variation of routing protocols (Source: [2])	12
2.3	LoRaWAN overview (Source: [3])	15
2.4	LoRaWAN class overview (Source: [3])	17
2.5	LoRaWAN receive windows (Source: [3])	18
2.6	LoRaWAN message format (Source: [3])	18
3.1	RREQ format	25
3.2	RREP format	25
3.3	RRER format	26
3.4	Data packet format	26
3.5	Routing table	27
3.6	Device table	28
4.1	RREQ data points for 1 gateway.	32
4.2	RREQ data points for 2 gateways.	32
4.3	RREQ data points for 3 gateways.	33
4.4	Successful amount of routes with different amount of gateways . . .	33
4.5	Average time for route construction with different number of gateways.	35
4.6	Number of packets needed for route construction with different number of gateways.	35
4.7	Experiment overview	36
4.8	Gateway placement during experiment	36
A.1	Graphical overview for tests	51

List of Tables

3.1	Routing commands	24
4.1	Delay time at gateways	31
4.2	Packets sent for each setup	34
4.3	Extra bytes transmitted for each setup	34
4.4	Experiment - route construction time	37
5.1	Construction time comparisons	40
5.2	Construction time comparisons in experiment	41
A.1	Lopy details	52

List of acronyms

AODV	Ad hoc On-Demand Distance Vector Routing
DSDV	Destination-Sequenced Distance Vector routing
DSR	Dynamic Source Routing
HWMP	Hybrid Wireless Mesh Protocol
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
LPWAN	Low Power Wide Area Network
MANET	Mobile ad hoc network
OLSR	Optimized Link State Routing Protocol
OSI	Open Systems Interconnection
RREP	Route reply
RREQ	Route request
RRER	Route error
WMN	Wireless mesh network
WSN	Wireless Sensor network
ZRP	Zone Routing Protocol

1.1 Background

The Internet of Things (IoT) is the idea that every object around us will be connected to the Internet. Often devices consist of embedded systems, which are hidden from the average human. IoT enables the possibility of smart homes, smart sensors and industry automation [4]. The IoT market is a rapidly growing market with many actors [5].

LoRaWANTM is a Low Power Wide Area Network (LPWAN) specification [3] intended for wireless battery operated Things in a regional, national or global network. LPWANs are designed to provide communication at low bit rate together with low power, compared to other wireless technologies designed to carry more data and therefore use more power. LoRaWAN targets key requirements of the Internet of Things such as secure bidirectional communication, mobility, and localization services. The LoRa Alliance [6] claims LoRaWAN provides seamless interoperability among smart things without the need for complex local installations and gives back the freedom to the user, developer, and businesses enabling the roll out of the Internet of Things. LoRaWAN is maintained by the LoRa Alliance [7] and figure 1.1 featured in a White Paper [1] from the alliance shows where LoRaWAN tries to fit in the myriad of IoT solutions.











	Local Area Network Short Range Communication	Low Power Wide Area (LPWAN) Internet of Things	Cellular Network Traditional M2M
	40%	45%	15%
	Well established standards In building	Low power consumption Low cost Positioning	Existing coverage High data rate
	Battery Live Provisioning Network cost & dependencies	High data rate Emerging standards	Autonomy Total cost of ownership
	 		   

Figure 1.1: LoRaWAN compared to other network solutions (Source: [1])

This thesis will discuss LoRa, LoRaWAN, wireless ad-hoc network, routing

and how this can be utilized by IoT to achieve mesh networks without the need of a permanent connection to the Internet.

Sensefarm [8] is a company focused on Agriculture 4.0 [9] which can be interpreted as IoT in agriculture. Sensefarm works in cooperation with local farmers' association and the local agriculture and technical universities. Their solutions are present in remote areas (such as farms and valleys) where there are disturbances in the GSM connection resulting in downtime of Internet connections, which is not suitable for IoT solutions. As an example, a farm uses surveillance applications to monitor temperature. The application's task is to send an alarm if there are changes in temperature. This system relies heavily upon a stable and always present connection and this is not feasible if there are no connections present. There is also loss of coverage and there is a need to connect repeaters to the existing LAN, this is today not supported by existing technology.

The aim of this Masters thesis work is to present a solution between LoRaWAN and LoRa gateways to provide network coverage and communication in remote areas. Today LoRa has gateways, which act as transparent bridges from end-devices to a network server. End-devices use single-hop wireless communication (radio) with gateways. Gateways connect to the network server via IP connections (Ethernet, 3G, Wi-Fi). The communications are bidirectional from end-device to network server. As described in the background, communication between the gateways and network cannot always be guaranteed. The aim of this thesis is to study the possibility of creating communication between the gateways, since they do not require a network to operate. We consider possibilities such as routing, mobile ad-hoc network and wireless sensor networks with related communication protocols.

1.2 Limitations

The thesis only analyses multi-hop technology and routing and their applications to the LoRaWAN stack. Radio is briefly touched on because of its tight relationship with LoRa and LoRaWAN but is not expanded upon. Neither is antenna placement discussed or presented but is a viable alternative to solve the problem with coverage.

Because of time limitations there is no possibility to develop a full mature and working network communications protocol. The development is limited to the parts that are needed to test vital functions of the protocol.

1.3 Method

A literature and system study was performed as an initial step to gain insight and in-depth knowledge of LoRa technology, multi-hop technology, and routing. Furthermore, studies were conducted on hardware platform used within the thesis. The study included both online work and scientific papers.

After the initial study phase a communication model was based on the knowledge gained from the previous step. The focus was on what models are needed and

what limitations and constraints are put on the model. Lastly part of the model was implemented, tested and evaluated.

1.4 A brief word about taxonomy

In this thesis, a wide range of terms will be presented and used. While studying the literature it became quite clear that the field is not consistent in the use of words and phrases. The choices of describing the same concept varies from article to article. The thesis will try to use the most used words and phrases found in references. If any ambiguity should arise, the thesis will try to mention alternative descriptions.

1.5 Related work

There are many related products that try to solve the same challenge as LoRaWAN is trying to solve. They are products aimed at solving IoT. They are all related to problem described in the introduction.

1.5.1 ZigBee

ZigBee is a wireless network standard that uses wireless mesh networks between wireless devices targeted at smart homes, sensor networks and general IoT applications. It is designed to provide low-cost and low-power devices and is maintained by the ZigBee Alliance [10]. ZigBee is built in on top of the IEEE 802.15.4 standard which defines low-rate WPANs.

ZigBee adds mesh networking too reduce the need of infrastructure and to increase coverage of the network.

1.5.2 SigFox

SigFox [11] is a company focused on delivering a full stack solution for IoT. It uses a star topology where every SigFox compliant device can send to deployed SigFox stations. The stations in return push all data to the SigFox Cloud where it is distributed to the customer. The goal is to provide low energy usage together with a lightweight communications protocol. It employs ultra narrowband radio combined with different modulation techniques on the open ISM bands [12]. The amount of data packets that can be sent during one day for each device is limited [11].

SigFox is one of the main competitors to LoRaWAN, they are built with similar models in mind trying to solve the same problem.

1.5.3 Google WiFi

Google WiFi is a smart home solution developed by Google [13]. The goal is to extend a home's current WiFi reachability without the need of more Ethernet connected devices. To achieve this each Google WiFi device can create a mesh

network. The mesh technology used is the IEEE 802.11s with undisclosed additions by Google [14]. The solution is similar to that of ZigBee but uses WiFi which has a shorter range than LoRaWAN.

1.6 Thesis outline

Chapter 2 presents the network technologies and theories used throughout the thesis, with a focus on multi-hop technologies and routing. Additionally LoRa and LoRaWAN is introduced.

Chapter 3 describes the network model and prototype that was developed.

Chapter 4 shows the testing procedure along with the results. Additionally a real-world experiment was performed and the results are presented.

Chapter 5 is a discussion and analysis of the experiment and testing.

Chapter 6 describes further work needed and concludes the thesis.

Network technologies

In this chapter, the technologies used in the thesis are presented. It starts with a brief introduction to wireless ad-hoc networks and the OSI model, then follows up with communication networks and routing in such networks. Afterwards it describes the LoRa and LoRaWAN protocol in detail and ends with a brief introduction to the hardware LoRaWAN is running on.

2.1 Wireless ad-hoc networks

A wireless ad-hoc network is a decentralized network that does not rely on pre-existing infrastructure. In an ad-hoc network, there are typically no routers or access points available. Instead, nodes rely on each other for communication. Each node that participates in the network is responsible for routing and forwarding data to other nodes. The topology of a wireless ad-hoc network can be highly dynamic because of mobile nodes and the network needs to adapt quickly to new scenarios [15].

2.2 Introduction to the OSI

The Open Systems Interconnection (OSI) model was proposed by the International Organization of Standardization in the 1970's. It is important to distinguish a model from a protocol suite. The OSI model is a seven-layered framework and is a model of the process of moving information across a network. The layers are, from bottom-up: physical, data link, network, transport, session, presentation and application. Every layer is responsible for completing the task given to it, before passing it on to the next layer below or above. The communication between layers is handled through well-defined interfaces. The focus hereafter will be on four of the layers: physical, data link, network and application. The physical layer is the lowest layer and responsible for carrying bits of data over a physical medium, for example USB or Bluetooth. The physical layer interfaces to the data link layer. The data link layer's responsibility is to transfer data between directly-connected nodes. It is often divided into two sublayers: Logical Link Control(LLC) and medium access control(MAC). LLC handles flow control, error notification and acknowledgment. The MAC sublayer can handle control of the medium e.g. CSMA/CD; it also contains the physical address of a device (MAC

address). Above the data link layer is the network layer, which is responsible for routing and host addressing; IPv4 operates at the network layer. At the top of the stack is the application layer. The layer consists of applications on-top of the stack, for example File Transfer Protocol(FTP) and Domain Name System(DNS). A graphical representation of the OSI model can be seen in Figure 2.1. An in-depth description of every layer can be found in most modern textbooks on computer networking. A good introduction can be found at [16].

OSI Model			
	Data unit	Layer	Function
Host layers	Data	7. Application	Network process to application
		6. Presentation	Data representation, encryption and decryption, convert machine dependent data to machine independent data
		5. Session	Interhost communication, managing sessions between applications
	Segments	4. Transport	Reliable delivery of segments between points on a network.
Media layers	Packet/Datagram	3. Network	Addressing, routing and (not necessarily reliable) delivery of datagrams between points on a network.
	Bit/Frame	2. Data link	A reliable direct point-to-point data connection.
	Bit	1. Physical	A (not necessarily reliable) direct point-to-point data connection.

Figure 2.1: OSI model (Source: IBM Knowledge Center)

2.3 Standards and amendments

The Institute of Electrical and Electronics Engineers(IEEE) has proposed and developed standards related to wireless multi-hop networking. 802.11s is the Wireless LAN(WLAN) mesh amendment to the 802.11 WLAN standard. The amendment requires that mesh network should appear in an Ethernet frame before and after entering a mesh network. Further, the standard defines that nodes within a mesh network are called mesh stations and communicate only with other mesh stations. Mesh capability is achieved through additions to the WLAN frame. Routing is performed with the Hybrid Wireless Mesh Protocol(HWMP). Since 2008 the 802.11s standard has been integrated into the Linux kernel through the `open80211s` vendor-neutral implementation. The amendment makes only minor changes to the MAC layer and can be implemented without any additional hardware or changes to existing network cards [17]. The 802.11s standard was superseded and integrated into the 802.11-2012[18], which in turn was superseded by the 802.11-2016 [19].

2.4 Wireless multi-hop network structures

In this section, four common ad-hoc network structures are presented. Wireless sensor networks(WSN), Wireless mesh networks(WMN), and Mobile wireless ad-hoc networks(MANET) are variations of Wireless ad hoc networks. They are all decentralized types of networks that do not rely on pre-existing infrastructure.

Delay tolerant networks(DTN) is a structure for overcoming unstable connectivity and long delays in network. These four are all studied in sections 2.4.1-2.4.4 because they try to solve the problem described in the introduction.

2.4.1 Wireless sensor networks(WSN)

A wireless sensor network [20] is a substantial number of distributed devices that communicate wirelessly and are often equipped with sensors for external measurement. All devices in the network work autonomously within the network. WSNs have a continuously changing topology, this is resulting from the nature of the network. Sensors or nodes within the network may for example become faulty because their batteries are exhausted. This changes the topology because data needs to be routed around a faulty node.

2.4.2 Wireless mesh networks(WMN)

Wireless mesh networks consist of mesh routers and mesh clients. Some or all nodes have a dual functionality as both host and router. A node will forward packets on behalf of another node that is not within wireless range of its target destination [15]. A typical WMN consists of several nodes of two types; access points(AP) and routers. Both routers and access points perform data packet forwarding but APs also serve end users. Some nodes may also act as gateway nodes towards a wired backbone e.g. the Internet [15][21]. Generally, WMNs are divided into three categories; Infrastructure/Backbone, Client or Hybrid. In an Infrastructure/Backbone approach mesh routers form a mesh backbone. Devices then connect to access points (also mesh nodes), which interfaces with one or several of the mesh routers. In the Client version, the mesh routers and access points are removed and the network consists of only the end devices. These form a peer-to-peer network where every node is both a router and operational application device. The client category is sometimes referred to as a mobile wireless ad-hoc network(MANET). In Hybrid mode, the last category, the network is a combination of infrastructure and client meshing. Devices may act in a local mesh client network but might also interface with a mesh router within a mesh backbone [15].

2.4.3 Mobile wireless ad-hoc networks(MANET)

Mobile ad-hoc networks(MANETS) are dynamically formed networks and consist of autonomous nodes. Nodes are not bound to a physical location and can move freely within the network. In a MANET, each node is equipped with wireless transmitters and receivers. Because of the free, unpredictable movement of nodes, the topology is constantly changing within the network [22][23]. The difference compared with WSNs is that the changing topology is due to movement between nodes, whereas in WSN it is usually due to nodes becoming faulty. MANET networks can operate autonomously in an isolated area without any preexisting infrastructure at the location [24]. The nodes themselves are the only infrastructure needed. The wireless medium used by the nodes often puts a constraint on the bandwidth capacity available for data packets. Very often, mobile nodes are not connected to a constant power supply but instead run on batteries; this puts

an energy constraint on the network [22]. MANETs are sometimes considered to be a subset of Wireless Mesh Networks due to their comparable properties and topology [15].

2.4.4 Delay tolerant networks(DTN)

Delay or Disruption Tolerant Networks are networks characterized by long delays and unreliable connectivity [25]. A model often used in DTNs to deliver messages is store-carry/keep-forward. The RFC [26] associated with DTNs describes the bundle layer, which is placed above the transport layer. The bundle layer is responsible for all operations that need to be performed, such as store, keep and diagnostics. If a node implements the bundle layer it is considered part of the DTN. Typical characteristics for DTNs are dynamic topologies, limited topological information, uncertain connectivity and limited available resources [25].

2.5 Routing in ad-hoc networks

A major issue in ad-hoc networks is routing. There is a large amount of research in the field focused on measurement and evaluation of different protocols depending on different characteristics [27]. In [27] a paper by Myung et al. they claim a routing protocol for wireless mobile ad-hoc networks consists of 5 core components and multiple auxiliary components. The core components are; route discovery, route selection, route maintenance, data forwarding, and route representation and metric. Route discovery is usually the first step when working with ad-hoc routing protocols. In this step, the purpose is to find routes towards potential or desired destination(s). Mobile ad-hoc routing protocols are at this step divided into three categories based on their characteristics: proactive, reactive or hybrid. This categorization is based on the topology information used by the protocols [28][29][30]. Different categories will be presented in detail in the next section because of their significant importance.

When route discovery is done, the next step is route selection. This can be done at source, destination or intermediate nodes but is generally divided into source selection or destination selection. When using source selection, the destination node replies to the source node with possible paths and the source decides which path to use. The same principle is used for destination selection but instead the path is chosen by the destination node. To be able to choose the best path, route representation and route metric are needed. Many different parameters can be chosen as route metric. Often the parameter is chosen so it can be minimized, for example shortest path, channel noise, hop count or latency. Route representation explains how the routing information is stored during route discovery and route selection.

2.5.1 Routing protocol categories

Wireless ad-hoc mobile networks are generally referred to as belonging to one of the below described categories because of the significant impact on how the routing protocol works internally [27].

Table-driven or proactive

In proactive routing, each node needs to continuously keep up-to-date routing tables and topology information for the whole network. When every node contains all information, they can each derive the most optimal route through the network. The nodes generally spread their network information in two ways: link-state or distance-vector. In link-state routing, each node updates its neighbors with the best-known distance to all other nodes. When using distance-vector routing, each node must inform all other node of the link cost of the node towards its neighbors. Updates are done periodically and all nodes distribute their routing tables to every other node in the network. Proactive protocols do this regardless of the traffic load in the network. Since the updates are performed at fixed intervals no consideration is taken of the current traffic or congestion of the network, nor is any consideration taken for a node's traffic requirements inside the network.

On-demand or reactive

Reactive routing protocols do not actively search and maintain routes in the network. They do not attempt to explore the network in any way until it's necessary. The need arises when a node receives or produces a data packet that must be sent. In a reactive protocol, there are two goals: find a route between the source and destination, and discover the optimal path. A reactive protocol has no knowledge of the network topology beforehand and floods the network with route request messages to find an available route.

Hybrid

Hybrid protocols try to combine the topology information of a proactive approach together with the route searching of a reactive approach. This can be used in specialized network scenarios. For example a network can have a part of the network being mobile where a reactive protocol delivers better performances. The rest of the network might be less mobile and a proactive protocol is better utilized.

2.5.2 Overview of common ad-hoc routing protocols

Each category described in the previous section contains several routing protocols. Due to the sheer number of them, a comprehensive overview can become complex and confusing [29]. The protocols chosen in this section are among the IETF experimental protocols, excluding the hybrid protocol. Additionally, Destination-Sequenced Distance Vector (DSDV) routing is chosen as it is commonly used and referenced in literature [21][27][29]. Furthermore, the protocols presented here are the most common routing protocols. In order of category they are: ad-hoc on demand distance vector (AODV), Dynamic Source Routing (DSR), Optimized Link State Routing Protocol (OLSR), Destination-sequenced distance vector (DSDV) and Zone Routing Protocol (ZRP) [29].

Ad Hoc On Demand Distance Vector(AODV)

The first reactive routing protocol is AODV explained in detail in [27][31]. AODV performs route discovery only when needed. It uses control messages such as Route Request(RREQ), Route Reply(RREP) and Route Error(RERR). A node starts route discovery by flooding the network with RREQ packets. RREQ packets do not store any information internally; instead each node that receives a packet stores information about the source, the destination, and the node that sent the packet. This information is used to set up a reverse path from the destination to the source node. When the RREQ reaches the destination node, or a neighboring node that knows the route to the destination, the node generates an RREP packet. The RREP packet is routed in reverse order back to the source node. With this process, each node caches information about nodes that might be inadequate or non-valid routes; the cache is discarded after a time interval. If a node moves, it sends out an RERR packet to all affected nodes. If a node that receives an RERR still needs the route, it can initiate a new route discovery with an RREQ packet.

Dynamic Source Routing(DSR)

DSR is a reactive routing protocol which has similarities with AODV and is explained in detail in [32]. Like AODV it also initiates the route discovery process by flooding the network with RREQ packets. The RREQ packet contains a list of hops which is collected. As the RREQ packet works its way through the network, each node it passes adds itself to the list of hops. This is done until the packet reaches the destination node. At the destination node, a RREP packet is generated and sent in reverse through the route based on the route provided in the RREQ packet. With this protocol, the source node might receive several RREP packets with different routes to the destination. The DSR protocol selects one of the routes, usually based on a defined metric such as shortest path or stability of the link. All routes are saved in a cache at the source node. By doing this the protocol can speed up the route discovery if the previously selected route becomes unavailable. Nodes which have already received a RREQ packet with the same identifier discard it to prevent RREQ packets circling around forever.

Optimized Link State Routing Protocol(OLSR)

The first proactive protocol presented is OLSR and is explained in detail in [33]. In OLSR, each node selects a set of its 1-hop neighbors to act as multipoint relays(MPR). The set of nodes chosen is selected such that it covers all symmetric strict 2-hop nodes. Symmetric nodes are nodes which have bi-directional linkage. A symmetric strict 2-hop neighborhood of a node is the set of nodes excluding the node itself and its 1-hop neighborhood which has a symmetric 1-hop neighbor and are not MPRs. If this is done correctly, every node in the network will have a symmetric link to an MPR. A node chosen as an MPR keeps information on all its neighbors who have chosen it as their MPR. When updating the routing information, every node broadcasts its information to its neighbors. Only nodes defined as MPRs will rebroadcast the message; neighbor nodes which defined as MPRs will simply not rebroadcast the information. The node updates are done

through periodic HELLO messages, thus a node can choose another set of nodes to act as MPRs. The use of MPR reduces the flooding of transmissions that occurs when the routing tables need to be updated.

Destination-Sequenced Distance Vector(DSDV)

DSDV is a proactive routing protocol and is explained in detail in [34]. The protocol is an improvement to the distance-vector routing protocol, which in turn uses the Bellman-Ford Algorithm. In the original distance-vector protocol, a table contains three fields: source, destination and number of hops required to get from source to destination. In the DSDV, one field, sequence number, is added to the routing table. Each node advertises its own routing table to all its neighbors both periodically and when triggered. The advertisement can be either unicast or multicast. When sending updates, the protocol can choose to use one of two methods to reduce the data carried over the physical medium: full dump or incremental carry. Full dumps are done at the periodic intervals and contain all available routing information the node knows about. Incremental carry is done when a node is triggered by another DSDV packet and has made an update to the routing table. This event only sends out the updated information the node has received. When a node receives a DSDV packet from another node it checks the sequence number inside the message and only updates its routing information if the message contains a higher sequence number from a node. If a node loses contact with a neighbor and thus removes the path to the node from its routing table it updates the sequence number and advertises the loss of the path to its neighbors. To summarize, each node maintains a routing table of all the other nodes it knows or has known about.

Zone Routing Protocol(ZRP)

The only hybrid routing protocol presented and it is explained in detail at [35]. In ZRP the network is divided into zones around each node. Every node has its own zone and the size is determined by the number of hops from the node. Within the zone, the Intrazone Routing protocol is used. Inside the zone each node uses neighbor discovery to find each node connected to it. The IARP uses a proactive protocol within the zone to keep up-to-date information about the local topology. If a node receives a data packet with a destination outside the zone, a reactive protocol is used to find a path. This is called Interzone Routing Protocol(IERP).

2.5.3 Alternate routing protocols

In addition to these protocols described above, there are many derivatives that mostly have their roots in one of the above presented protocols. Examples include radio aware AODV. These protocols may further be grouped into geographical, geocast, multi-path, power aware etc [2][27][28][29][30]. Figure 2.2 is an overview of [2] showing the multitude of routing protocols.

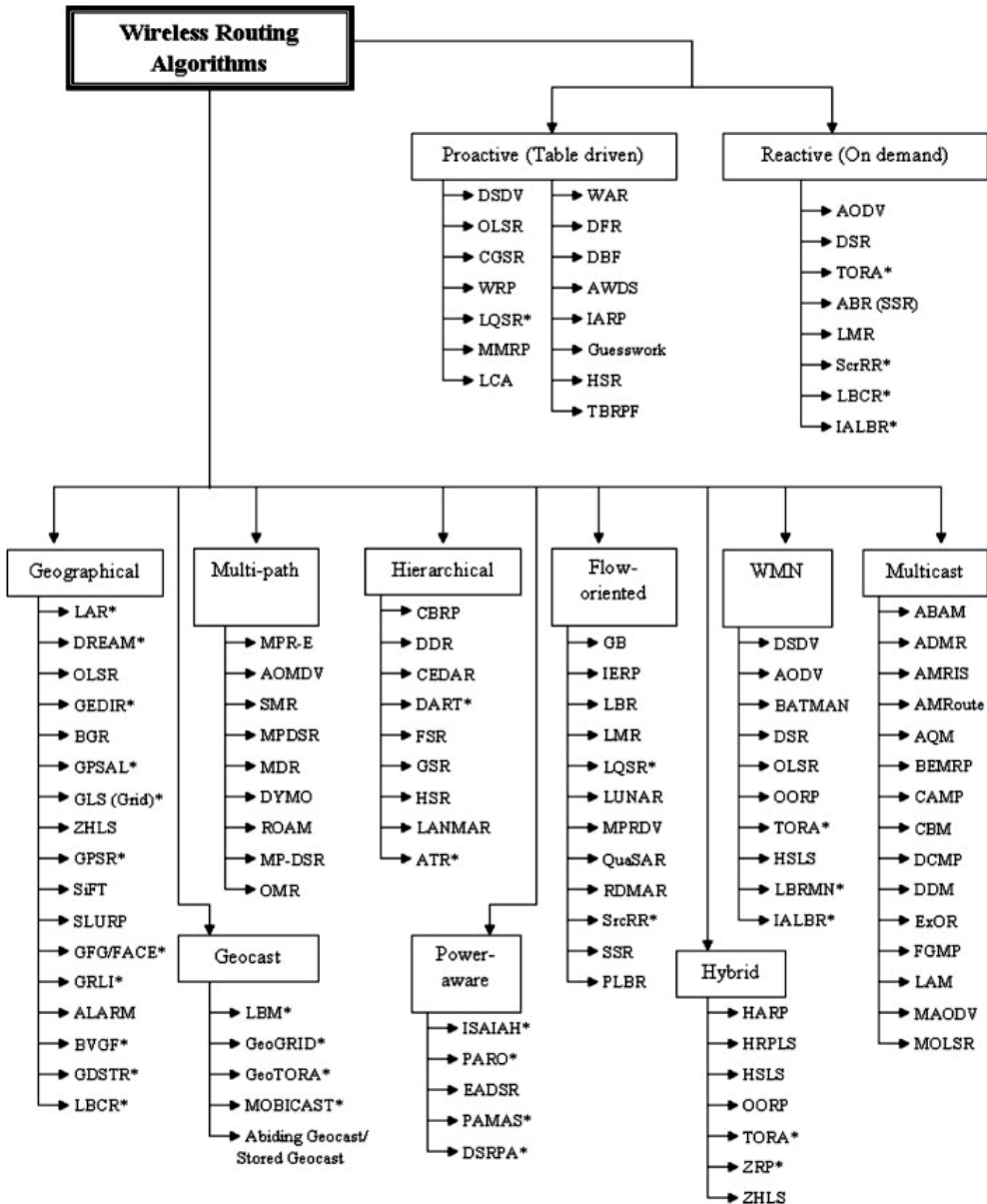


Figure 2.2: Different variation of routing protocols (Source: [2])

2.5.4 Performance and characteristics

Performance and characteristic analysis is often done in simulators such as NS-3 [36], QualNet [37] and OMNET++ [38]. However, some evaluation can be done by the nature of the protocol. An initial evaluation of the categories to which the protocols presented above belong immediately gives some characteristics of the protocol. A proactive protocol stores a large amount of data at each node and thus requires continuous maintenance to keep the data up to date. The protocol needs to continuously send data across the network to update data. However, the routing information at every node is directly available [21]. A reactive protocol does not store data locally for a long time and does not need to actively send data across the network. Instead it needs to flood the network with data at certain times. The routing information is not directly available unless still present as cache data in the node [21]. To summarize based on categories: a proactive protocol has a low delay in transmission but a larger number of packets being sent across the network, whereas a reactive protocol has significantly bigger delay but a lower number of packets sent.

Throughput and average delay for DSR, AODV and ZRP have been studied using NS-2 [21]. The authors' results show that AODV has the highest throughput compared to the other protocols. This remains unchanged as more nodes are added to the network. Regarding average delay, the authors conclude that AODV and ZRP have a higher average delay. They attribute this to the nature of the protocols. AODV is reactive and creates routes when needed and ZRP does the same as the number of nodes increases. In a similar study also using NS-2 [39], the author obtained similar results for both throughput and average delay. In their studies, they use more nodes and initially DSDV performs better than AODV but the protocol's performance converges as the number of nodes increase.

Lastly the RFC of each protocol gives an overview of its suggested applicability. RFC 3526 OLSR [33] suggests that OLSR is well suited for large and dense networks because of the optimization done by the MPRs. It scales well with a larger and denser network compared to a classic link state algorithm. Furthermore, they suggest the protocol is well suited for networks where traffic is random and sporadic between a large set of nodes.

In RFC 4728 DSR [32], it is suggested that DSR is designed for networks of up to 200 nodes and works well with high mobility rates. RFC 3561 AODV [31] suggests that AODV is designed for tens to thousands of nodes. The protocol can handle a variety of mobility rates as well as a variety of data traffic levels.

2.5.5 Routing in Wireless Mesh Networks

Wireless Mesh Networks(WMN) were presented earlier with different setups. IEEE 802.11s [17], released in 2012, is an amendment for mesh networking to the IEEE 802.11 standard [19][40]. It defines mesh networking on top of wireless local area networks(WLAN). The 802.11s standard proposes a routing protocol called Hybrid Wireless Mesh Protocol(HWMP), which is a hybrid ad-hoc routing protocol. The protocol is based on a combination of AODV and tree-based routing [40], where Tree-based routing is where a node stores the routing information in tree structure. There exist different variations of tree structures [41].

HWMP can be run in two different modes depending on the topology and layout of the network: on-demand or proactive. Reactive mode is used if there is no root node configured within the network. In this mode, it uses a radio-aware type of AODV. In proactive mode, a root node is configured for the network. The root node periodically sends routing information and metric information downwards through the network. In this mode all nodes learn a route towards the root node which acts as a bridge/interface to another network [40].

Four control messages are used within the HWMP: Path Request(PREQ), Path Reply(PREP), Path Error(PERR) and Root Announcement(RANN). PREQ, PREP and PERR work almost identically to their counterparts in AODV. HWMP extends the functionality of PREQ in proactive mode. The root node periodically sends proactive PREQ downwards with the destination field set to broadcast (all set to one). The proactive PREQ is processed at every node and then rebroadcast; through this process every node knows a way towards the root node. Another difference from AODV is the addition of proactive RANN messages; their purpose is to update route metrics to the root. RANN messages are periodically sent through the network and every node must reply with a RREQ sent towards the root node. The root node in return answers with a PREP to the node.

In a study [42] performed with the NS-3 simulator comparing AODV and HWMP, it was shown that HWMP outperforms AODV in packet delivery fraction, throughput and end to end delay.

2.6 LoRa/LoRaWAN

LoRa technologies are divided into two sub technologies which are tightly knitted together: LoRaTM and LoRaWANTM. LoRa is the radio technology used to communicate and LoRaWAN is the protocol for the communication. LoRaWAN is proposed as an infrastructure solution for the Internet of Things. LoRaWANs are laid out in a star-of-stars topology. There are three main components used within the specification: end-devices, gateways and a network server. End-devices use wireless single-hop communication to gateways. Gateways relay data to the network server using a standard IP connection over Ethernet, 3G or WiFi. The end-to-end communication is bidirectional in most cases; there is support for both multicast and one way messages. The full LoRa protocol stack is based on the use of both LoRa and LoRaWAN. LoRa works at the physical layer, using regional ISM bands for communication. Above the physical layer is the MAC layer where LoRaWAN operates.

2.6.1 LPWAN

LoRaWAN is considered to be part of Low Power Wide Area Network(LPWAN) category[1]. In a white paper released by Link Labs in 2016, LPWAN features are long range, low data rate and low power consumption [43]. Generally LPWAN operates at a lower frequency than other wireless technologies and at low power, but this comes at a cost of the amount of data that can be sent in a given time.

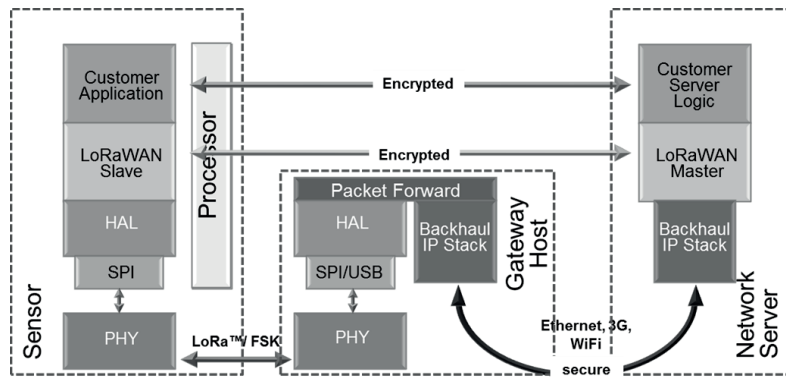


Figure 2.3: LoRaWAN overview (Source: [3])

2.6.2 LoRa

LoRa is the physical layer of the full protocol stack. It is developed by Semtech and is still their proprietary technology. LoRa is an acronym for Long Range and is a modulation technique that can achieve long range communication. Modulation in LoRa is based on a variation of chirp spread spectrum and forward error correction. Because it operates only on the physical layer it can be used in several protocol stacks[44]. Depending on the transceiver LoRa can achieve a data range between 0.018 and 38.4kpbs[44].

2.6.3 LoRaWAN

The LoRaWAN standard is defined in [3]. In the following sections are features presented in the standard and not a full coverage. LoRaWAN operates at the data link layer, that is, directly on top of the physical layer provided by LoRa. While LoRa's specification is proprietary, the LoRaWAN specification is open and currently developed by the LoRa Alliance[7]. The LoRaWAN specification defines several components to form a LoRaWAN network: End-devices, Gateways and a network server. End-devices are often low power sensors, for example temperature sensors. End-devices communicate with gateways using LoRa. Gateways are transparent relays that work as intermediate devices. Gateways forward packets from an end-device to a network server. This is done using standard IP connections over Ethernet, 3G or WiFi. Several can be present in the same LoRaWAN. Different gateways can also receive the same data packet from a single end-device. The network server is responsible for decoding and dealing with duplicate packets. It can receive the same data packet from several gateways and needs to remove duplicates. It is also responsible for generating a packet to be sent back to an end-device through one of the gateways.

The end-devices are not bound to a gateway for delivering their message. When an end-device intends to send, it broadcasts its data packet, which is received by any gateway listening. All gateways then forward their packets to a single network server. Gateways add a layer of information describing the reception quality of the communication link. This information together, with the original data packet,

is then sent to the server. The network server needs to detect duplicate packages and eliminate the unwanted ones. It generates a new packet to send back and calculates which gateway had the best reception. This gateway is then chosen for relaying back data to the end-device; for a graphical overview see Figure 2.4. Depending on the needs of an application, the specification defines three different classes of end-devices: Bi-directional end devices (Class A), Bi-directional end-devices with scheduled receive slots (Class B) and Bi-directional end-devices with maximal receive slots (Class C) .

In Bi-directional end devices (Class A) each uplink transmission is followed by two short downlink receive windows. The transmission slot is scheduled by the end-device when it needs to send data. This means that the end-device decides when data should be communicated between the server and the end-device. The server has no way of sending until it receives data from the same end-device again. This is the class that has the lowest power consumption. The disadvantage is the inability for the server to send data downstream at will, it must cohere to the end-device receive windows.

The Bi-directional end-devices with scheduled receive slots (Class B) opens up for more receive slots. It implements class A's random time slots and adds extra scheduled receive windows. For this to work the gateway needs to send time synchronization messages(Beacons). These tell the server when the end-device is listening.

Bi-directional end-devices with maximal receive slots (Class C) are the closest to continuous communication that can be achieved within the protocol. Devices in this class constantly keep their receive windows open; the receive window is only closed when transmitting data. Of the three classes, it is easy to see that this is the most power consuming implementation. The advantage is the low latency achieved. Class B and C must have class A functionality but because of the nature of the class C format it cannot contain an implementation of class B and vice versa. Furthermore, all end-devices must implement class A. All classes therefore share the same physical message format. When the protocol described hereafter it will be described from a class A perspective with addition of B and C when they differ or add functionality.

Physical message

In the LoRaWAN specification, there is a distinct difference between uplink and downlink messages. Uplink messages are sent by end-devices to the network server through one or many gateways. A downlink message is sent by the network server to one end-device through a single gateway. At the physical layer an uplink message consists of the following fields: preamble, PHDR, PHDR_CRC, PHY Payload and CRC (also shown in Figure 2.6).

PHDR is the LoRa physical header and PHDR_CRC is the CRC for the PHDR field. PHY Payload contains the data sent to the MAC layer. The ending CRC is calculated for the whole message. The downlink message has the same physical message format but message CRC is removed to reduce the data sent out on the ISM band.

After each uplink transmission, the end-device can receive data. This is ac-

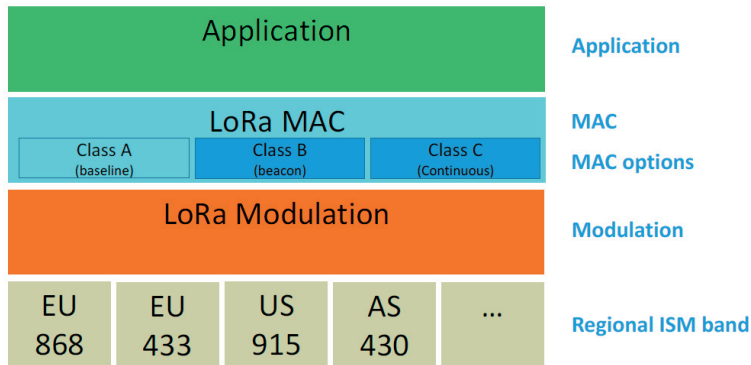


Figure 2.4: LoRaWAN class overview (Source: [3])

completed by the end-device always opening two short receive windows, RX1 and RX2, after every uplink transmission. The start times for RX1 and RX2 are defined by the regional parameters provided by the LoRa-Alliance and vary between different regions. For Europe, RX1 is set to start after 1s and RX2 after 2s. These intervals are timed by the end-device, which starts the calculation after the end of an uplink transmission. If any preamble is detected during the receiving window time, the window stays open until the receiver has received the full message. The length of a receive window is required to be the time it takes for the end-device radio transceiver to detect a downlink preamble. Consequently, the receiver will not open its second receive window if it identified a packet intended for itself during the first window. If a network server intends to send a downlink transmission, it starts the transmission at the beginning of either RX1 or RX2. An end-device is not allowed to send another uplink transmission before it has either received a downlink message in RX1/RX2, or the RX2 window has expired. End-devices may cooperate with other protocols or perform any form of transmissions outside of the windows and still be compatible with the specification, as long as they are compatible with local regulations.

MAC message

In every uplink and downlink message, a PHYPayload field is featured inside the physical frame. After extracting the data from the physical frame in the previous layer, it arrives at the data link layer. The content of the PHYPayload field varies depending on what features of the protocol is used. The first field MHDR and the last field MIC are always present. MHDR is the MAC header and contains version information about the LoRaWAN protocol used. MIC field is the message integrity code for the frame. The middle field can be one of three possible fields: MAC Payload/Join-Request/Join-Response. Join-Request and Join-Response is used for activation of end-devices (see below). MACPayload is used in all other cases.

The MACPayload in turn consists of three more fields: FHDR, FPort and

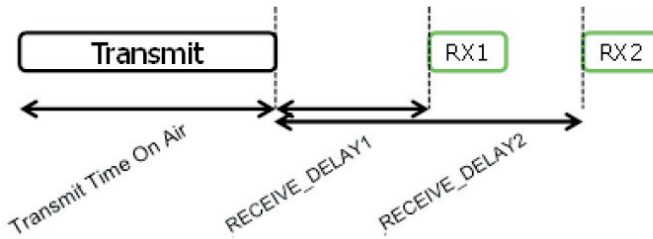


Figure 2.5: LoRaWAN receive windows (Source: [3])

FRMPayload. FRMPayload contains either data payload from the end-device or MAC commands (see below). FPort must be present if the FRMPayload field is present. The FPort value can be set if the payload needs to be sent to a specific application; the exception is value 244, which is reserved for testing, and value 0, which indicates that the FRMPayload consists of only MAC commands.

The last field to expand is the FHDR. It contains three more mandatory fields and one optional field. The first field in FHDR is the DevAddr; it contains the short version of the device address. The next field is FCtrl and its field slightly varies if it is being sent via the uplink or downlink. It contains control of adaptive data rate, pending bit if more data is available at the server, and an ACK field if ACK is required by any device. Further along the FHDR is the frame counter FCnt; it tracks the number of frames being sent between the end-device and the server. Lastly the FHDR has the FOpts field which can contain MAC commands.

Radio PHY layer:

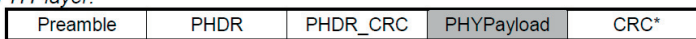


Figure 5: Radio PHY structure (CRC* is only available on uplink messages)

PHYPayload:

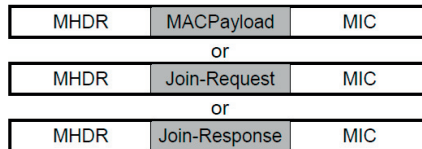


Figure 6: PHY payload structure

MACPayload:



Figure 7: MAC payload structure

FHDR:

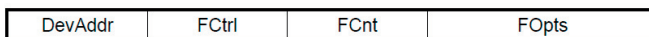


Figure 8: Frame header structure

Figure 2.6: LoRaWAN message format (Source: [3])

MAC commands

LoRaWAN has built-in commands for network administration called MAC commands. They are communicated between the network server and the end-device. MAC commands can be sent independently or be piggybacked in the FOpts field or in the FRMPayload. The current specification defines 18 commands, which appear in pairs, one request and one answer. Administrative tasks that can be performed are for example connectivity check or changing receive slot parameters. A complete list can be found in the specification.

End-device setup

To able to participate within a LoRaWAN network, all end-devices must be activated and personalized. Within the protocol this can be done in two ways: Over-The-Air-Activation(OTTA) or Activation By Personalization(ABP). When OTTA is used, the Join-request/Join-response is used to share keys over the network. If ABP is used the end-device and network server need to be pre-programmed with all keys. When the activation is complete four parameters are stored in the end-device; a device address(DevAddr), an application identifier(AppEUI), a network session key(NwksKey) and application session key(AppSKey). The first two parameters, DevAddr and AppEUI, are used to identify an end-device within a network. The DevAddr parameter is a 32-bit field that identifies a specific end-device. AppEUI is an ID parameter, which follows the regulations of the IEEE EUI 64 [45] address space. The session keys NwksKey and AppSKey are used for integrity and security. NwksKey is specific for each end-device and is used by the server as well to calculate the MIC in all messages. Lastly the AppSKey is used by both the device and server as an encryption and decryption key.

Payload size

The LoRa technology utilizes the regional open ISM bands. The amount of data that can be sent is limited by the physical layer. It depends on the effective modulation rate and on if the FOpt field is used or not. In the EU, the 863-870 MHz band is used and depending on the data rate the maximum payload available ranges from 51 to 222 bytes.

Gateways

As presented earlier gateways are the transparent bridges in the LoRaWAN protocol. They are responsible for data transport between end-devices and the network server. As described in the introduction, end-devices use LoRa to broadcast their data. Any number of gateways in range of an end-device can receive the transmitted data and forward it to the network server. The gateways are completely unaware of all other gateways; one exception is when class B devices are used where gateways are promoted to Beacons. Then there is some form of communication between the gateways. Gateways communicate with the network server over a standard IP stack. This is done via Ethernet, 3G, WiFi or GSM. The result

is that gateways perform very few operations and act as repeaters and translators within the network. In production, each gateway needs to have a network server address set. Gateways do not drop any packets or sort packets it receives. Gateways are identified on the network through MAC addresses.

LoRa Packet forwarder

The gateway communicates upstream with a server through a simple UDP protocol, which is described in detail at [46]. A summary is presented hereafter. When the gateway receives a packet from an end-device, it records statistics regarding the transmission, time, frequency, RSSI etc. It encapsulates these statistic into a JSON format together with the payload received from the node and sends this data to the server through a `PUSH_DATA` command. The server responds with a `PUSH_ACK` command. Downlink communication is initiated by the gateway by sending a `PULL_DATA` command telling the server it is ready to receive packets. The gateway initiates the communication because it might be located behind a firewall or NAT. A `PULL_ACK` is sent from the server to confirm an open downlink. `PULL_DATA` must be sent in periodically to inform the server that the gateway is still listening. When the server wants to send data downstream it sends a `PULL_RESP` containing the downlink payload and JSON packet with information about transmission time and LoRa parameters. Lastly the gateway will respond with a `TX_ACK` packet for every `PULL_RESP` to acknowledge a successful downlink transmission to the gateway. `TX_ACK` packets might contain error codes if something went wrong.

LoRa in the development environment

An important aspect is the development and production environment in which a network operates. At Sensefarm sensors are often buried or in other ways obstructed. The result is a lack of line of sight between end-devices and gateways. Within an operational area, the number of gateways are limited but there are many sensors. Data is sent as JSON files between end-devices and network server.

2.7 Implementation device

Implementation and coding have been done on a LoPy 1.0 manufactured by Pycom [47]. The LoPy is a microcontroller that supports LoRa, WiFi and Bluetooth. The LoPy can also act as a “nano-gateway” as described by Pycom. The term “nano” refers to the LoPy’s inability to act as a LoRaWAN gateway described in the specification. According to the specification a gateway needs to be able to listen to at least three separate channels whereas the LoPy can only manage communication on a single channel [3]. However, it has been certified as an end-device by the LoRa-Alliance [48]. Running on the LoPy is MicroPython [49], a C implementation of the Python 3 programming language designed for microcontrollers.

Before starting this chapter let us revert to the original problem: the problem was to be able to extend a LoRaWAN network for better connectivity. The proposed solution in this thesis is to adapt a LoRaWAN network to a multi-hop network. The chapter starts off with considerations within different areas, followed up by the selected wireless structure with modifications needed for LoRaWAN and lastly the proposed solution with implemented features is presented.

3.1 Considerations

A number of things have to be considered when designing a routing protocol for LoRaWAN. First and most importantly it must not interfere with the already existing implementation. It is vital that all original functionality is preserved for at least two reasons. First, there exist several devices which are already conforming to the LoRaWAN protocol and support the latest implementation. Secondly a good protocol should not interfere with lower level layers or preexisting protocols if not explicitly needed.

3.1.1 Hierarchy

A major difference between the LoRaWAN topology and an ad-hoc network is the strong hierarchy enforced within the LoRaWAN star of star topology. In ad-hoc there is no hierarchy by default, though some extensions and implementations with hierarchy have been proposed and developed [2]. The hierarchy within LoRaWAN can be viewed from two perspectives, from the end-device or from the server. If the hierarchy structure is viewed from the end-device perspective they can be viewed as masters in the network because they effectively control the congestion and data flow without knowing it. Since the protocol is set up to almost only facilitate data from an end-device to the server they unbeknown to themselves become the masters of the network. If viewed from the server side it contains the LoRa master as described in the specification. All end-devices are slaves because they are unaware of their surroundings. The server is the only entity in the network aware of the topology and decides about downstream transmission. The gateways are stuck in the middle as messengers between the two masters and only relays

data. One exception is when class B devices are used, then the gateways act as beacons and sync the time in their local star.

3.1.2 Data limitations

Because of the limitations of the LoRa technology there is a limitation of how much data can be sent in one message. The LoRaWAN packet without a data payload is at least 13 bytes depending on the usage. Payloads can vary between 59 and 230 bytes depending on region as described in the specifications.

When developing a routing protocol within LoRaWAN this puts a constraint of how much data can be sent and used by the protocol. The number of bytes used by standard routing protocols strongly exceeds that of a couple of bytes that is normally transmitted by LoRa.

3.1.3 Time limitations

As described earlier there is a strict time constraint on the receive windows, 1 and 2 seconds for a round trip time of data. This means that a proposed routing protocol must be able to handle routing fast and deliver data within the given time constraint or keep an already established routing table available.

3.1.4 Duty cycle limitations

LoRaWAN is operating in the open ISM band 868 MHz which enforces a 1% duty cycle. The original protocol with a star of star topology is designed to minimize the data sent over LoRa. A routing protocol must adhere to the same limitation and not impose a significantly bigger load.

3.1.5 Addressing

Each end-device is given a unique DevEUI and DevAddr from the server. The server is reached through either hostname or IP-address. Gateways are not designated anything but their own MAC address.

3.1.6 Surrounding layers

Most routing protocols are built on higher layers in the OSI model. A consequence of this is that there is no TCP/IP stack in the LoRaWAN protocol. A further consequence of this is that a routing protocol built upon LoRaWAN needs to address some issues caused by the absence of, for example, the IP header. There is no source and destination IP address available for peer-to-peer transmission, neither is there any retransmit functionality.

Furthermore, it is not a viable option to implement a bigger stack because of the data constraint imposed by the LoRaWAN protocol. For example, the IP header alone adds 20 bytes as a minimum, which is too much to start with. It's also designed for bigger datagrams up to 65,535 bytes and not for minimal data.

3.1.7 Production environment

The last consideration is regarding to the operational environment the application is going to be used in. Each gateway can be fitted with GSM, WiFi, Ethernet or any other technology available to transmit data to the Internet. This gives two options for possible routing: root gateway with a stable connection or finding a gateway with a connection available.

The gateways in a network are never going to be a large number. They have can receive data over long ranges and few are enough to provide coverage even in large areas. This results in a smaller network of gateways.

3.2 Suitable wireless multi-hop technology

Taking into consideration the multi hop technologies presented earlier together with the considerations presented above, the goal is to minimize data overhead and end-to-end time. The operating network is small and will not contain many nodes. Starting with the topology overview, the topology of HWMP resembles the topology of the production environment. It has both proactive and reactive features which are directly applicable to the gateways environment. It can be selected for both root mode, and for a smaller network it can use AODV.

The next step would be to look at data overhead and timing; neither a proactive nor reactive protocol are alone able to beat both low overhead and minimal delay times. Both factors need to be minimized to work well within LoRaWAN. AODV performs on average well in delay times compared to the other protocols but is a reactive protocol and requires a big amount of data overhead to construct routes at every occasion. This shortcoming is fixed in HWMP by using the proactive features and thus minimizing the need for route constructions. However, HWMP uses bigger protocols and unnecessary features not always needed in LoRaWAN where's AODV uses less information.

As result of this section a good protocol/model could be based on HWMP/AODV.

3.3 Proposed model/protocol

In this section, a complete model for ad-hoc integration with LoRaWAN is presented. With the considerations taken above; the selected methods is a tunneling protocol which borrows its features from AODV and HWMP. The model follows the features from AODV and HWMP if nothing else is specified. A full protocol is not presented, only basic features and enough features to perform initial testing.

3.3.1 Overview

The proposed model is to implement an ad-hoc network on the gateways and allow them to act as mesh nodes within a LoRaWAN network. When a gateway receives a packet from an end-device it checks if it has network, if that is the case it carries as specified in the LoRaWAN specification. If it has no connection it can start mesh networking in two separate ways. If a master gateway(root mode) exists it

forwards the packet to the master gateway. If no master exists it perform a route lookup for a gateway that has connection.

Ad-hoc features are done strictly between gateways thus the end-device and network server are unaware of any routing within the network. It can be viewed as tunneling from a gateway, receiving data from an end-device, to the gateway transmitting to the server.

The general principle and features are built on top of the AODV and HWMP protocols. The command messages are modified to work with LoRaWAN and to minimize the data needed to send.

Recognition of a ad-hoc LoRaWAN packet is done through the original MAC header. The three fields are specified as MType, RFU and Major. The MType is set to *111* which in the specification translates to Proprietary which is reserved for non-standard message formats. The RFU field is reserved for future use and within it we set it to *001* to uniquely identify the message as an ad-hoc packet. Lastly the Major field is set to *00* as is all other messages within a LoRaWAN. The original header is kept so the packet can be recognized as a LoRaWAN packet on the network, this can be helpful if some entity within the network keeps a statistic record of packets. After the header the ad-hoc network commands are added to identify which type of packet follows thereafter, a complete list is presented in table 3.1.

Table 3.1: Routing commands

Command	Identifier
Data packet	0x01
RREQ	0x02
RREP	0x03
RRER	0x04

3.3.2 Addressing

Gateways within the network use their 8-byte MAC address to identify themselves. The end-devices also have a MAC address in our case but they are never advertised within the protocol. So, to reduce data needed to send to the gateways instead use the 4-byte long Device Address to identify end-devices. The gateways address is also referred to as node id throughout the text.

3.3.3 Route request

The RREQ command used in the model can be viewed in figure 3.1. They only fields that have been kept are hop count, RREQ id, destination id, destination sequence number, originator node id, originator sequence number and metric. An additional field Source node id has been added. The source node id is needed since there is no IP layer or any other addressing method present in the protocol. It contains the node id of the previous node in the chain. The destination node id is

set to all ones to indicate a broadcast and any node can answer if it has a Internet connection.

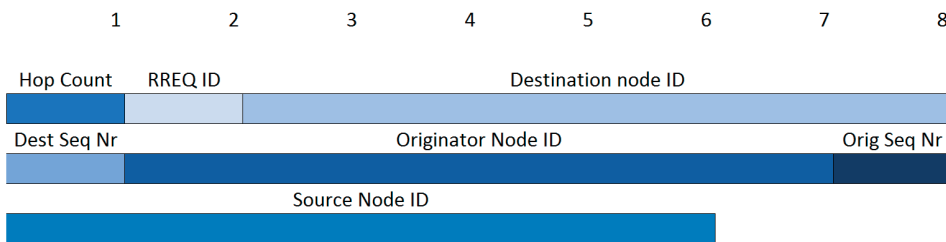


Figure 3.1: RREQ format

3.3.4 Route reply

The RREP command used in the model can be viewed in figure 3.2. Fields related to dependent mesh points have been removed to lower the amount of data transmitted over the network. Dependent mesh points are neighboring points, which have a route to the root through the local mesh point. They are not needed for the initial step of the protocol but might be needed if a full proactive implementation of HWMP is done. As with RREQ the TTL field has been removed since it currently fills no function in the implementation, however it might be needed in a later stage to avoid congestion and cycling. The same goes with the length and flag fields. The problem with a missing IP-layer is present in the RREP packet as well, therefore an additional field for the source node id is present.

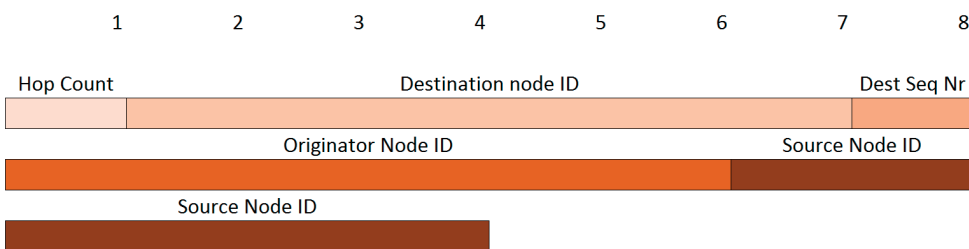


Figure 3.2: RREP format

3.3.5 Route error

The RRER command is almost unchanged compared to the RRER used by AODV. It already contains all the fields required. A graphical overview can be seen in figure 3.3.

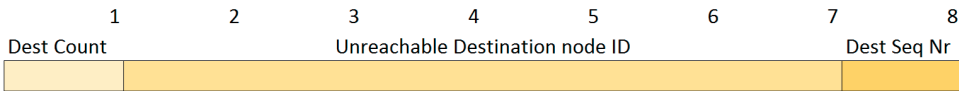


Figure 3.3: RRER format

3.3.6 Data packet

A data packet is sent from a gateway when it has a route to a connected gateway. In front of the original data packet the destination node and source node are applied. The size of the data and JSON packet can vary in size. A graphical overview can be seen in figure 3.4. The question marks in the figure is due to the variable size of the data packet and the JSON file that is transmitted.

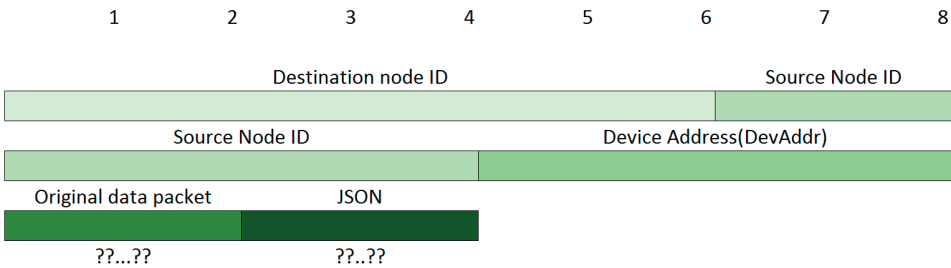


Figure 3.4: Data packet format

3.3.7 Routing tables, device tables and buffers

To be able to perform the routing each gateway needs to store routing information, device table and buffers.

Routing table

Each gateway keeps tracks of the routing information in a routing table. Each entry stores 5 variables: Destination node ID, Next hop node ID, Destination Sequence Number and hop count. A graphical overview can be seen in figure 3.5.

Destination node id Holds the value of the destination node in the routing entry

Next hop node ID Node that holds a route to the destination node

Destination sequence number Holds the sequence number

Hop count Contains the hop count to destination node id

Routing tables are updated on almost the same conditions as in AODV and HWMP. Routes are updated based on two conditions: destination sequence numbers and hop count. The first condition, destination sequence number, is used to always maintain a valid and stable path. A route will always be updated on a higher destination sequence number. If a node receives a packet with the same destination sequence number it will continue and look at the other condition: hop count. If a packet is received with the same destination sequence number but a lower hop count, the node will update its routing table to maintain the shortest possible path to the originator.



Figure 3.5: Routing table

Keeping track of end-devices

Gateways need to keep track of the end-devices who can reach them. Each time a gateway receives an uplink messages it stores the address of the end-device in an array. When it later on receives a downlink message it does a lookup if the end-device is reachable from the gateway otherwise it proceeds to forward it to the next gateway. This works because downlink messages are only transmitted after an uplink message, so a gateway will not forward a message intended for one of its end-devices.

Device tables

When a network connected gateway receives an ad-hoc routed packet it sends it to the network server. According to the LoRaWAN stack the server will use the same gateway (if only one received it) for downlink transmit. Because of this the gateway needs to keep track of which gateway to forward the downlink transmission. This is done using a lookup-table, when the gateway receives a packet with itself as destination it registers this into its lookup-table. The table saves two values; Device address(**DevAddr**) and destination node id.

Device address(DevAddr) Contains the Device Address of the end-device the packet was received from

Destination node id Node id of the gateway originating the ad-hoc routed packet.

When the gateway received the downlink ötransmission, it does a lookup and matches the device address and sends it to the next hop route for the received destination id. A graphical representation can be found in 3.6.

Description of network flow

This section will give an overview of the flow from when an end-device needs to send until it has acquired a message back.

Device table

**Figure 3.6:** Device table

- The first thing that happens is that an end-device wakes up and transmits its message, it then proceeds to await the opening of its first receive window.
- A gateway is constantly listening and receives the message transmitted by the end-device. If root mode is used it checks if it has a valid path to the root through its routing table, if such a path exists the gateway transmits the packet further. If no such path exists or root mode is not used it does a path lookup. This is done through the following steps:
 1. The gateway sends out a RREQ message with the destination address set to broadcast and awaits a RREP message.
 2. When a RREP message is received the gateway stores the routing information about the destination into its routing table.
 3. After storing the routing information, it proceeds to store the DevAddr of the end-device in the lookup list.
 4. As a last step the gateway compiles together the data packet and transmits it.
- The next gateway in the network receives the transmission from the originator and checks if it is the destination. If its not the destination it looks into its routing table for a valid route and transmits the packet if such a path exists and updates the Source Node ID. If its the destination it performs the following steps:
 1. It stores the DevAddr and the Destination Node ID into its device table to keep track for the downlink message.
 2. Decompile the data packet into the original LoRaWAN message.
 3. Sends the original LoRaWAN message to the server and awaits response.
 4. If no response is received within a time limit it erases the entry in the device table. If a response is received from the server the gateway performs a lookup in the device table to get the destination address corresponding to the DevAddr obtained from the server and erases the entry.
 5. The gateway compiles together the data packet with the same JSON data from the original packet and adds the destination address. It performs a lookup into the table to see if it still has a valid route to the destination.
 6. Transmits the packet.

- The next gateway in downlink chain checks if it is the destination (originating gateway) and has the DevAddr stored in its lookup table, if it is not the destination it looks into its routing table for a valid route and transmits the packet if such a path exists and updates the Source Node ID. If its the destination it performs the following steps:
 1. Erases the DevAddr entry in the lookup table.
 2. Decompiles the data packet into a LoRaWAN complaint packet and transmits it to the end-device.

Experiment, testing and evaluation

The purpose of the tests conducted is to measure some of the requirements imposed by the LoRaWAN specification. To be able to meet the time constraints of the receive windows, time tests are done with ad-hoc routing and without ad-hoc routing. During the test, packet loss is measured from end-to-end and the amount of data packets sent are recorded. This can be used to calculate the overhead data the ad-hoc protocol adds. LoRaWAN specifies the 1% duty cycle and thus it's important to measure the overhead.

4.1 Testing

4.1.1 Test setup

For the tests five LoPy units were used. One LoPy acted as an end-device within the network. One acted as the final gateway sending to the server. The three remaining LoPy's acted as intermediate gateways in the network. The Open Source LoRa server [50] acted as the receiving server. Because of the shared frequency shared by all the devices a time delay is added before each gateway sends its packet. Delays were chosen arbitrary with an interval of 0.1 seconds and tested between two devices. 0.1 second was the lowest time where the devices could still send messages to each other without disturbing the transmission. The delay of each gateway can be seen in table 4.1. Delays could be eliminated if a multichannel device was used for communication. A multichannel device could setup specific channels to other devices to reduce collisions. Furthermore a multichannel device would be able to listen on multiple channels, which will not block antennas when transmitting. For a detailed and graphical test setup see appendix A.

Table 4.1: Delay time at gateways

Gateway	Originator	First	Second	Destination
Delay time(s)	0.1	0.2	0.3	0.01

4.1.2 Constructing a routing path

Building a routing path is the method where the originator gateway sends a route request(RREQ) out on the network requesting a path towards the destination. The construction ends when the originator node receives a route reply(RREP) from the destination node. The time starts when a gateway finds out it needs a get a route and ends when it has processed the RREP packet. The test was conducted with 200 iterations. This was done with 1,2 and 3 gateways between the originating gateway and server. The results are shown in figure 4.1, 4.2 and 4.3. For each graph the amount of data points with corresponding time is plotted together with the 95% confidence interval.

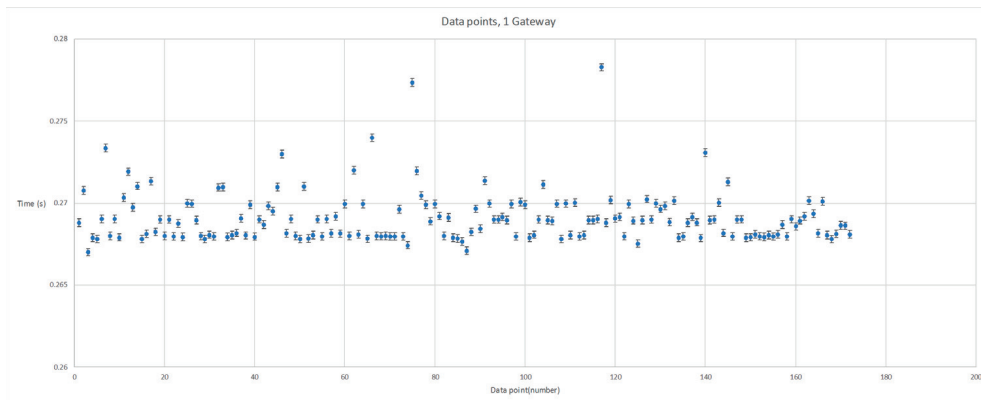


Figure 4.1: RREQ data points for 1 gateway.

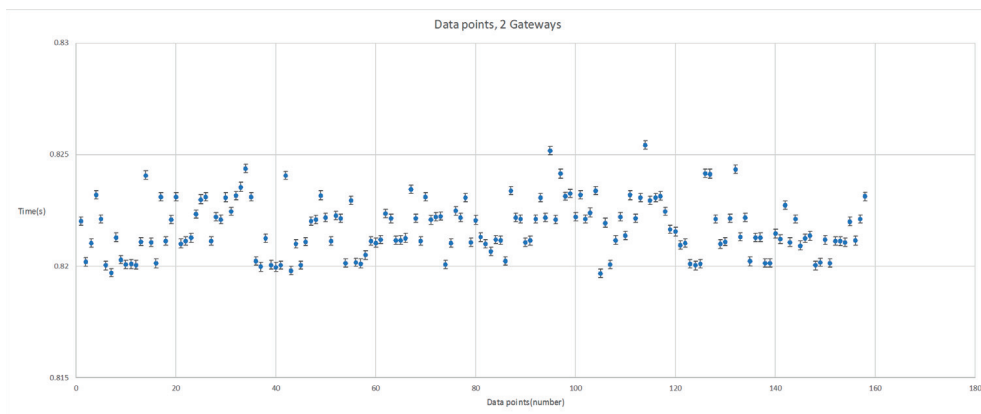


Figure 4.2: RREQ data points for 2 gateways.

Successful routes achieved

During the previous test the number of successful routes were also tested. The result is shown in 4.4.

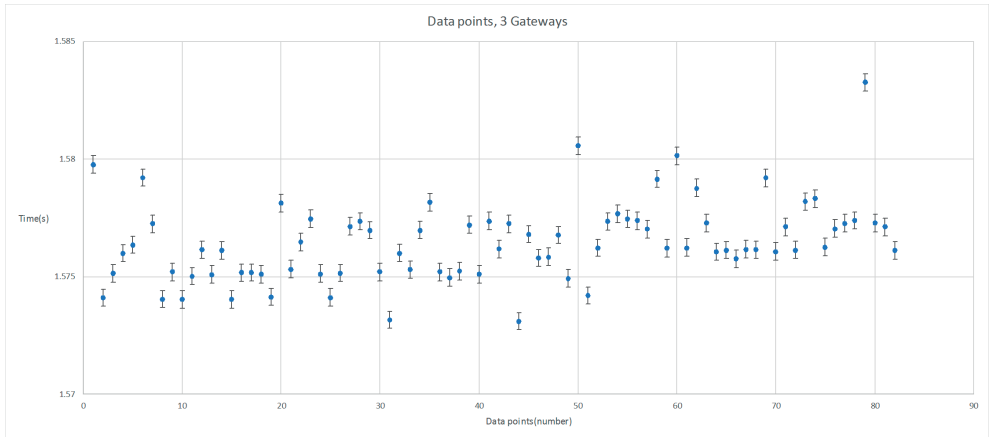


Figure 4.3: RREQ data points for 3 gateways.

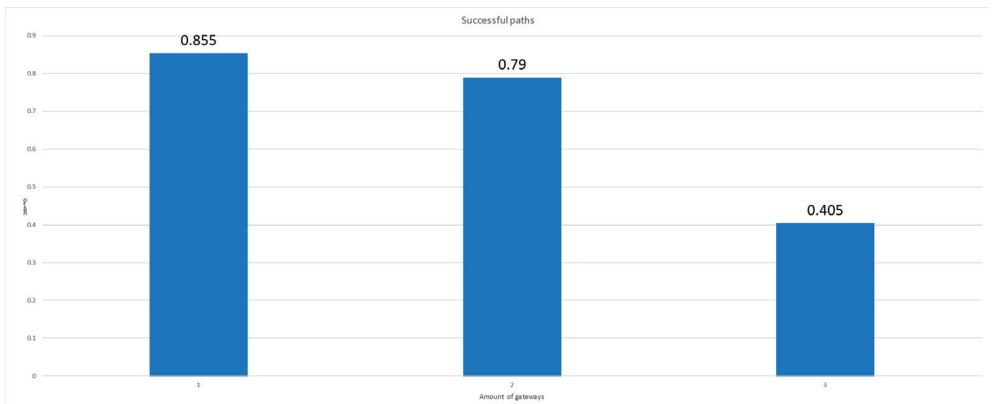


Figure 4.4: Successful amount of routes with different amount of gateways

4.1.3 Data load

Two different calculations can be performed on the data load. The first is the number of packets sent out for each additional gateway and can be calculated by the formula where G denotes the number of gateways:

$$1 + G \times 2$$

The result is shown in table 4.2. The two packets are the RREQ and RREP sent out by each additional gateway in the test setup. The single packet in the beginning of the formula is the packet beings sent from the end-device to the gateway.

The second calculation is the extra bytes transmitted with each addition of a gateway. Its given by the formula;

$$G \times 44$$

Where 44 is the total number of bytes for one RREQ packet and one RREP packet. If RREQ and RREP size changes a more accurate formula would be:

$$G \times (RREQ + RREP)$$

where RREQ and RREP is the size of each of the two packets. The result is shown in table 4.3.

Table 4.2: Packets sent for each setup

Amount of extra gateways	0	1	2	3
Number of packets	1	3	5	7

Table 4.3: Extra bytes transmitted for each setup

Amount of extra gateways	0	1	2	3
Extra bytes transmitted	0	44	88	132

4.1.4 Comparison and overview of data

Figure 4.5 and 4.6 shows summaries and overviews of the data.

4.2 Experiment

Experiments with the implementation were done at a customer location. Part of the original problem was to extend the coverage of LoRaWAN. Sensefarm has a customer which needed sensor data from within a warehouse. Tests conducted by Sensefarm revealed that existing gateways could not provide full coverage inside the warehouse. For their test they used one or two Kerlink Gateways [51]. The

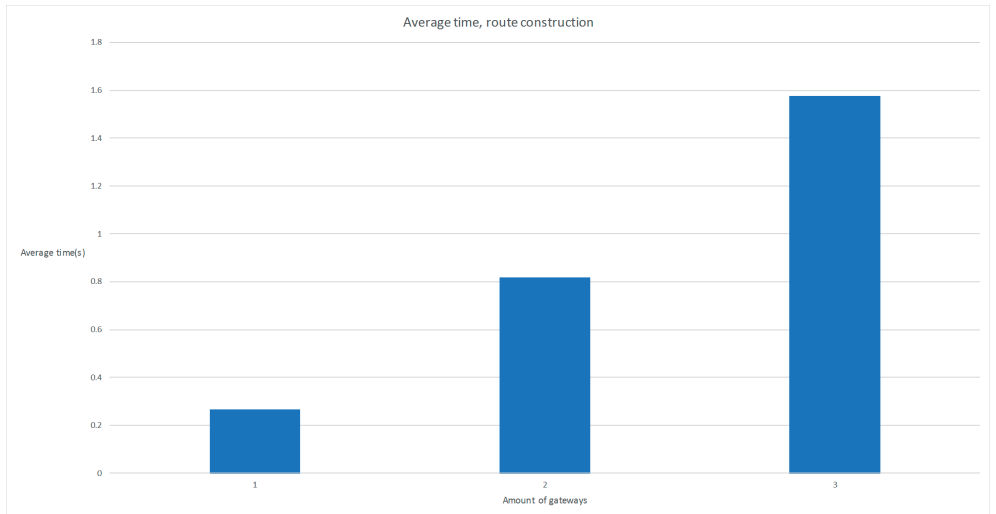


Figure 4.5: Average time for route construction with different number of gateways.

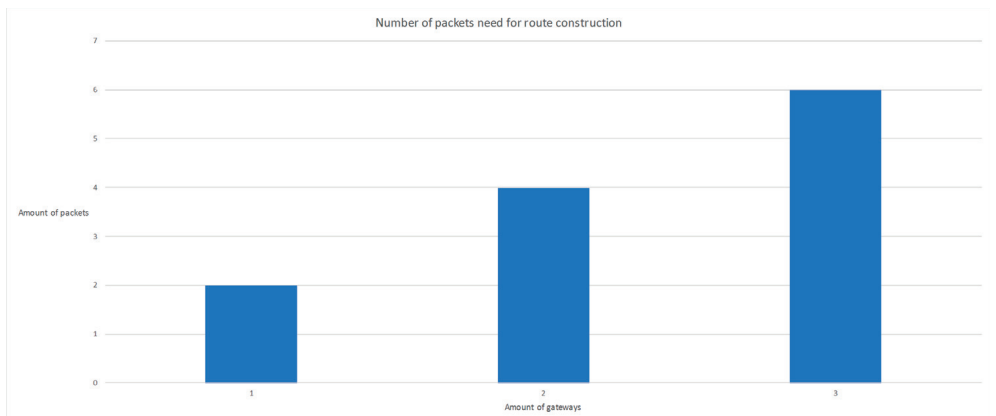


Figure 4.6: Number of packets needed for route construction with different number of gateways.

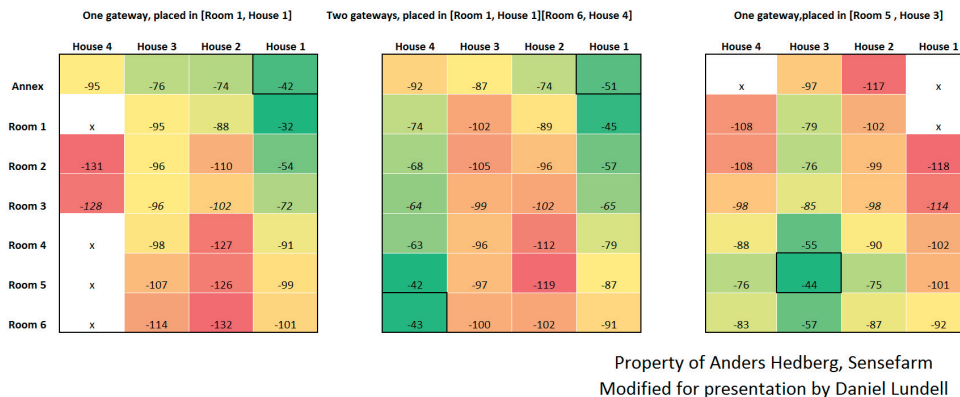


Figure 4.7: Experiment overview

RSSI given in *db* for each section was recorded. Sections with no reception is marked with an *x*. Gateway placement is marked with black borders. The result is shown in figure 4.7.

In the same location, the same tests as described in the previous section were performed; the construction of a routing path. This test used an older implementation but with the same principles as the one described in this paper.

4.2.1 Test setup

Three LoPy devices were placed as shown in figure 4.8. They performed a construction of a routing path. Tests were performed with one and two extra gateways.

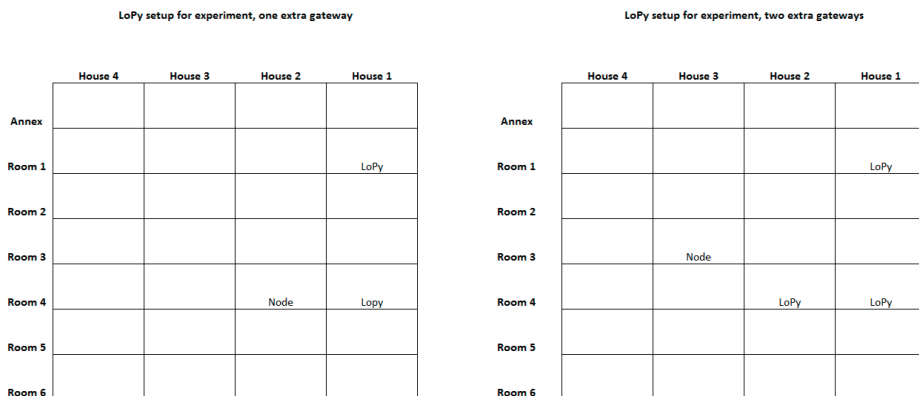


Figure 4.8: Gateway placement during experiment

4.2.2 Result

Each test was only performed three times. The resulting times are shown in table 4.4.

Table 4.4: Experiment - route construction time

Number of gateways	1	2
Test 1	0.327943	0.889042
Test 2	0.328961	0.887943
Test 3	0.327911	0.880154
Average	0.328271	0.889046

5.1 Timing

For the protocol to be a viable option it must be able to meet the timing constraints imposed by the specification. The test does not present an optimal scenario because of the fixed frequency but still gives a strong estimation. The first note, is that the tests were not performed with more than 3 extra gateways. Still a strong linear trend to the increased time needed to perform routing can be seen. As can be seen in the graphs for each test the data points don't deviate very much. This trend can be seen in all time tests performed. It hints of a deterministic execution for both LoRa and MicroPython.

For the test to work properly delays had to be added to the gateways for the packets to not collide in the air. The tests used the LoPy, but testing with another device with multichannel capability might have resulted in lower construction times. If they could send on multiple frequency and internally setup different frequencies between them they can send and listen simultaneously. If this was the case there would be no need for a delay time. However, this might increase the processing time when a network grows large because of the share amount of data each gateway's internal processor has to process.

The test does not include the actual data transfer after a route has been established but the test still shows problems at two extra gateways. Two extra gateways give a response time of roughly 0.85 seconds which is already close to the deadline for RX1 window of 1 second. At three extra gateways the time is up to an average of 1.5 seconds which is fast closing in on RX2 window of 2 seconds. Delay times must be considered, the third gateway adds 0.6 seconds in delay time. The second gateway adds 0.4 seconds. Construction time without delays can be seen in table 5.1. If we remove the delays we get an increase of approximately 0.15 seconds for each device added to the network. The remaining time is the transmission time of LoRa and code execution time. If multichannel devices can be used the depth of the network graph could be increased to roughly 6 devices and still fit into the time limits. This calculation does not take network congestion into consideration but calculation times can be assumed to be small because of the relatively small network packets.

Table 5.1: Construction time comparisons

Total delay	Average time, with delays	Average time, no delays
0.11	0.2690956	0.159095623
0.51	0.8213547	0.3113647
1.11	1.5761935	0.4661935

5.2 Data overhead

In a standard LoRaWAN setup each uplink communication only needs to transmit one LoRa packet. Shown in the test at 4.2 each gateway adds to the length of the chain 2 additional packets required for a transmission. Shown in table 4.2 is the extra number of bytes transmitted for each additional gateway when constructing a route. It has a linear increase which probably can be attributed to the test setup which only tests a straight line and not a complete mesh. The LoRaWAN adds at least 13 bytes and random data loads. Just compared to the header the extra bytes transmitted is of factor 3.5 higher. This is a significant increase and it is increasing linearly as more gateways are added and is quickly lots of factors higher.

5.3 Successful routes

In figure 4.4 the ratio of successful routes are shown for each number of extra gateways. At one and two extra gateways the successful routes are still low but quickly rises as a third gateway is added. It's a significant decrease from roughly 85% to 40%. The result is not very surprising because more packets need to be successfully delivered in the network. This increases the probability of packet loss due to noise or fading resulting in less packets being delivered successfully. The test was only performed once with 200 iterations and further testing is needed to determine if that was an anomaly or if it's a persistent effect. Furthermore, more tests with more gateways are needed to get a complete insight.

5.4 Experiment

The experiment carried out at an actual location was an interesting field study. The limited time made an impact on the number of tests carried out and the number of iterations it could be run. As described in the experiment an earlier version of the routing protocol was used. They had the same functionality but varied in how the fields were used. This resulted in a larger number of bytes sent out on the network and slower execution time. Even if few data points were obtained some remarks to the result can be done.

The results hints of a deterministic increase in time as were seen in the test carried out multiple times. They however show a 0.06 seconds slower time. This is likely the result of the earlier protocol. In table 5.2 the delay times are removed,

Table 5.2: Construction time comparisons in experiment

Total delay	Average time, with delays	Average time, no delays
0.11	0.328271	0.218271
0.51	0.889046	0.379046

the difference is approximately 0.16 seconds which is close to the results achieved in the tests.

5.5 Integrating directly into LoRaWAN protocol

An earlier draft of the proposed protocol contained a different approach to solving the integration of ad-hoc and LoRaWAN. The solution was still based on AODV/HWMP but tried to extend the MAC commands feature used within LoRaWAN instead of tunneling. The MAC commands version had some problems. The LoRaWAN payload had no concept of remembering the gateway to which it arrived, it only contained the device address of the node sending it. Addition of MAC commands were easy because the specification allows the use of proprietary commands and some commands are reserved for future use and was thus accessible to use. With this approach fields would be left blank such as; FCtrl and FCnt. The device address and the gateway MAC address are different in length and could not be switched out efficiently. This left unnecessary unused bytes in the protocol and was switched out for a protocol that uses all bytes available.

Future work and conclusion

This paper does not define a complete routing protocol nor does it provide solution to all the problems encountered. A lot of enhancement and further work can be done, hereafter they are presented in a categorical overview and at the end a conclusion.

6.1 Routing

A problem with the solution is the loss of packets. This might be a problem only occurring on the chosen development platform and might be solved by utilizing other methods such as multichannel devices.

The proposed solution suggests the use of hop count as the link cost or link budget for all routes within the network. This is the most straightforward metric to use for minimizing data sent over the network. Another method to select route is to use a probabilistic approach, for example based on the probability of congestion in the network. Every node can record its traffic and can based on probability choose the least trafficked path. Another solutions is to use Received signal strength indication(RSSI) to determine the path with lowest noise and interference to increase successful delivery of packets.

LoRa technology can change spreading factor and frequency to increase or decrease both transmission range and data rate. This can be incorporated between gateways; each gateway analyzes its surrounding and selects best spreading factor and frequency to the next gateway or even reach further to the next gateway in range.

Not implemented in the solution is a feature of AODV which might reduce the data traffic being sent. It is the possibility for a node neighboring the destination node to send an RREP instead of forwarding it. This was not implemented and not tested but will probably reduce the data load on the network.

6.1.1 Route Errors

Route error management is presented in the solution but is not fully studied with regards to LoRaWAN. AODV and HWMP specifies criterion in their specifications but more factors for errors can be found and some can be removed. Because of the relatively non-mobility of gateways some route error factors are eliminated.

Instead route errors might occur when a gateway loses connection to the Internet thus becoming inoperable.

6.1.2 Route timers

In both the AODV and HWMP standard routes become obsolete after a time. This is not featured in the protocol described in the thesis. A simple solution is to follow the standard timers used in AODV and HWMP but this is probably not optimal for LoRaWAN. The gateways are very stationary and won't move thus eliminating factors that requires construction of a new route. The more advanced solutions are to further study the probability of downtime on gateways and set timers after those results.

6.1.3 Downlink transmission

Downlink transmission is not implemented in the solution. The gateway adds information about the transmission and appends it to the less costly UDP packet. The packet is almost always the same and adds additional bytes to the transmission. This makes the total transmission larger and might not be optimal. As can be seen in the test the additional bytes required to perform routing already makes the cost of transmission a lot higher. The JSON file added to the data packet contains data that is needed for the downlink transmission to work correctly, for example it contains the timestamps used by the gateway to sync with receive windows. In this solution the packet is just sent back and forth, an alternative solution would be to store it at the gateway together with the DevAddr and only extract the important parts and transmit them instead of the whole JSON file.

6.1.4 Acknowledgement

Due to time constraints, there was not enough time to add acknowledgement or retransmission features. Acknowledgements need to be done for both data packets and routing packets, neither exists in the implementation. In most protocols the problem would be solved by the TCP/IP stack. In the current implementation, there exists no form of acknowledgement except the RREQ/RREP packets. In a full, mature and working protocol some sort of acknowledgement should be implemented to guarantee the delivery and aid in route error detection.

6.1.5 Unicast to multicast

The protocols discussed throughout the thesis are described from a unicast perspective. Inside LoRaWAN the natural communication becomes multicast because of the long range that is achievable by LoRa. Also, radio waves can reflect and interfere with packets already in the air increasing the difficulty of decoding them. Some of these issues are fixed with the implemented data buffers in the solution. The data buffers store the messages and limit the chance of resending packets that are already sent. However, the implementation does not handle a node sending to multiple gateways which in turn start route requests. In the LoRaWAN stack this is handled at the server side and is possible because packets received from

the same node via different gateways will arrive within a short time frame. In the ad-hoc solution they must first construct routes and then send the data. As a first problem, it increases the data amount several magnitudes if several gateways perform route lookup for the same data packet generating a large overhead. Neither is there a guarantee that the packets will arrive within the same time frame. A full protocol needs to deal with this to avoid multiple reports of the same data, avoid congestion and minimize the data sent over the network.

6.2 Security

Security is not offered by the proposed solution. These issues are addressed in the HWMP protocol. The system is very vulnerable to man-in-the-middle attacks of the route selection. LoRaWAN offers two way of registering end-devices; ABP and OTAA. When ABP is used the security, keys are shared beforehand and never sent out on the network. If this method is used an attacker won't be able to decrypt the payload. OTAA uses the pre-shared AppKey, which uses an AES-128 key to encrypt and decrypt. If an attacker can break the encryption the network might be vulnerable. Furthermore, an attacker with knowledge of the protocol could be able to reroute packets to a desired destination since no encryption is done on the routing data. This is a problem that needs to be addressed before a viable solution can be produced.

6.3 Conclusion

This thesis has research into multi-hop technology and tried to apply the methods learned onto LoRaWAN. A basic model/protocol has been developed for testing if multi-hop is a feasible solution. The tests and experiment show the possibility of a working ad-hoc network implementation for LoRaWAN. The construction times quickly raises as more gateways are introduced because of the delays, with no delays a network with a depth of six nodes is possible without breaking the time limit. LoRaWAN can modify the windows and increase them to fit a larger network. If that solution is used it would require a back-end server with a deep knowledge of the networks topology to manage all gateways. This in turn would require custom written algorithms based on top of LoRaWAN and more complex computing at the server side. The same problem arises with the packet collision, a good knowledge of the topology could control gateways and set them to internally use different frequencies.

A bigger problem is the data overhead that comes along with more advanced routing, the rule imposed by the duty limit can rapidly be broken as the networks increase in size. The topology will play an important rule of how quickly the duty cycle limit will be crossed. A spread out network might more slowly reach the limit while a dense network might reach it faster. In a tree topology with more nodes closer to the root it might be broken because of more traffic going through nodes closer to the root.

Tests showed a big decrease in successful routes achieved as more gateways where added. Further research with fully featured mesh networks should be con-

ducted to establish if this a trend. Additionally congestion of a full mesh network was not concluded and should be made.

LoRaWAN might not be the optimal candidate for ad-hoc routing because of its drawbacks described earlier in the thesis.

However, if a full modification of a protocol like HWMP can be implemented for LoRAWAN there is a possibility to achieve a result in smaller networks where there is less traffic present. At Sensefarm, a fully functional protocol could completely make an Internet connection unnecessary, solutions could be made to work autonomously with only LoRaWAN. Furthermore it could provide mesh networks with a very large coverage with minimal infrastructure needed. It might not perform well within city or urban areas where a lot of radio transmission already occur. LoRa can already be used to form mesh networks [44] but LoRaWAN adds the infrastructure needed for an IoT solution. This crosses into the domains of GSM and 4G, which can already manage great distances of communication. LoRaWAN with mesh capability has the methods to concur with the distance but not the data rate.

The solution presented in the thesis will solve the problem described in the introduction if the protocol can be fully developed. The developed model lacks functions to fully work and operate. It is important to develop a working testbed with complete mesh network for further tests. Further tests should be conducted to investigate out the possibility of a working LoRaWAN ad-hoc network. If it can be achieved there is a possibility to build networks with very long range, covering large areas.

References

- [1] “Lorawan white paper.” <https://www.lora-alliance.org/portals/0/documents/whitepapers/LoRaWAN101.pdf>. (Accessed on 05/07/2017).
- [2] E. Alotaibi and B. Mukherjee, “Survey paper: A survey on routing algorithms for wireless ad-hoc and mesh networks.,” *Computer Networks*, vol. 56, pp. 940 – 965, 2012.
- [3] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, “LoRa alliance LoRaWAN specification,” *LoRaWAN Specification, Release v1.0*, 2015.
- [4] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, “Internet of things (iot): A vision, architectural elements, and future directions,” *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645 – 1660, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services - Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.
- [5] “Roundup of internet of things forecasts and market estimates, 2016.” <https://www.forbes.com/sites/louiscolombus/2016/11/27/roundup-of-internet-of-things-forecasts-and-market-estimates-2016/#7d24e199292d>. (Accessed on 05/28/2017).
- [6] “Lora technology.” <https://www.lora-alliance.org/What-Is-LoRa/Technology>. (Accessed on 04/25/2017).
- [7] “Lora alliance.” <https://www.lora-alliance.org/>. (Accessed on 05/07/2017).
- [8] “Sensefarm develops products for precision farming..” <http://www.sensefarm.com/>. (Accessed on 05/04/2017).
- [9] “What is "agriculture 4.0"? | cema - european agricultural machinery.” <http://www.cema-agri.org/page/what-agriculture-40>. (Accessed on 05/04/2017).
- [10] “What is zigbee? | zigbee alliance.” <http://www.zigbee.org/what-is-zigbee/>. (Accessed on 05/04/2017).
- [11] “Sigfox technology overview | sigfox.” <https://www.sigfox.com/en/sigfox-iot-technology-overview>. (Accessed on 04/27/2017).

- [12] "Radio technology keypoints | sigfox." <https://www.sigfox.com/en/sigfox-iot-radio-technology>. (Accessed on 04/27/2017).
- [13] "How it works – google wifi – made by google." <https://madeby.google.com/wifi/how-it-works/>. (Accessed on 05/03/2017).
- [14] "Making a 'mesh' of your wi-fi." <https://blog.google/products/google-wifi/making-mesh-your-wi-fi/>. (Accessed on 05/03/2017).
- [15] I. Akyildiz and W. Xudong, "A survey on wireless mesh networks.," *IEEE Communications Magazine*, vol. 43, no. 9, pp. S23 – S30, 2005.
- [16] W. Stallings and M. M. Manna, *Data and computer communications*. Harlow : Pearson, cop. 2014, 2014.
- [17] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke, "Ieee 802.11 s: the wlan mesh standard," *IEEE Wireless Communications*, vol. 17, no. 1, 2010.
- [18] "Ieee sa - 802.11-2012 - ieee standard for information technology–telecommunications and information exchange between systems local and metropolitan area networks–specific requirements part 11: Wireless lan medium access control (mac) and physical layer (phy) specifications." <https://standards.ieee.org/findstds/standard/802.11-2012.html>. (Accessed on 06/03/2017).
- [19] "Ieee 802.11, the working group setting the standards for wireless lans." http://www.ieee802.org/11/Reports/802.11_Timelines.htm. (Accessed on 04/25/2017).
- [20] C. Guy, "Wireless sensor networks," in *Sixth International Symposium on Instrumentation and Control Technology: Signal Analysis, Measurement Theory, Photo-Electronic technology, and Artificial Intelligence*, pp. 63571I–63571I, International Society for Optics and Photonics, 2006.
- [21] P. Kavita and S. Abhishek, "A comprehensive performance analysis of proactive, reactive and hybrid manets routing protocols.," *International Journal of Computer Science Issues*, Vol 8, Iss 6, Pp 432-441 (2011), no. 6, p. 432, 2011.
- [22] S. Corson and J. Macker, "Mobile ad hoc networking (manet): Routing protocol performance issues and evaluation considerations", rfc 2501," 1999.
- [23] M. Conti and S. Giordano, "Mobile ad hoc networking: Milestones, challenges, and new research directions.," *IEEE COMMUNICATIONS MAGAZINE*, vol. 52, no. 1, pp. 85 – 96, 2014.
- [24] I. Chlamtac, M. Conti, and J. J.-N. Liu, "Mobile ad hoc networking: imperatives and challenges.," *Ad Hoc Networks*, vol. 1, no. 1, p. 13, 2003.
- [25] Z. Feng and K.-W. Chin, "A survey of delay tolerant networks routing protocols.," 2012.
- [26] "Rfc 4838 - delay-tolerant networking architecture." <https://tools.ietf.org/html/rfc4838>. (Accessed on 05/05/2017).

- [27] M. Lee, J. Zheng, X. Hu, H. Juan, C. Zhu, Y. Liu, J. Yoon, and T. Saadawi, "A new taxonomy of routing algorithms for wireless mobile ad hoc networks: The component approach.," *IEEE COMMUNICATIONS MAGAZINE*, vol. 44, no. 11, pp. 116 – 123, 2006.
- [28] G. Walikar and R. Biradar, "A survey on hybrid routing mechanisms in mobile ad hoc networks.," *Journal of Network and Computer Applications*, vol. 77, pp. 48 – 63, 2017.
- [29] I. Ahmad, U. Ashraf, and A. Ghafoor, "A comparative qos survey of mobile ad hoc network routing protocols.," *Journal of the Chinese Institute of Engineers*, vol. 39, no. 5, pp. 585 – 592, 2016.
- [30] K. Chawda and D. Gorana, "A survey of energy efficient routing protocol in manet.," in *2nd International Conference on Electronics and Communication Systems, ICECS 2015*, no. 2nd International Conference on Electronics and Communication Systems, ICECS 2015, ((1)Computer Science and Engineering, Parul Institute of Engineering and Technology), pp. 953–957, 2015.
- [31] "Rfc 3561 - ad hoc on-demand distance vector (aodv) routing." <https://tools.ietf.org/html/rfc3561>. (Accessed on 04/25/2017).
- [32] "Rfc 4728 - the dynamic source routing protocol (dsr) for mobile ad hoc networks for ipv4." <https://tools.ietf.org/html/rfc4728>. (Accessed on 04/25/2017).
- [33] "Rfc 3626 - <https://www.rfc-editor.org/rfc/rfc3626.txt>." <https://www.rfc-editor.org/rfc/rfc3626.txt>. (Accessed on 04/25/2017).
- [34] C. E. Perkins and P. Bhagwat, "Highly dynamic destination-sequenced distance-vector routing (dsv) for mobile computers," in *ACM SIGCOMM computer communication review*, vol. 24, pp. 234–244, ACM, 1994.
- [35] "draft-ietf-manet-zone-zrp-04 - the zone routing protocol (zrp) for ad hoc networks." <https://tools.ietf.org/html/draft-ietf-manet-zone-zrp-04>. (Accessed on 05/05/2017).
- [36] "ns-3." <https://www.nsnam.org/>. (Accessed on 05/04/2017).
- [37] "Network simulator software - packet trace." <http://web.scalable-networks.com/qualnet-network-simulator-software>. (Accessed on 05/04/2017).
- [38] "Omnet++ discrete event simulator - home." <https://omnetpp.org/>. (Accessed on 05/04/2017).
- [39] P. Manickam, T. G. Baskar, M. Girija, and D. D. Manimegalai, "Performance comparisons of routing protocols in mobile ad hoc networks.," 2011.
- [40] "Hwmp specification." <https://mentor.ieee.org/802.11/public/06/11-06-1778-01-000s-hwmp-specification.doc>. (Accessed on 07/05/2017).

- [41] "Tree-based routing protocol for mobile wireless sensor networks.," *2011 Eighth International Conference on Wireless On-Demand Network Systems and Services, Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on*, p. 164, 2011.
- [42] "Performance comparison of aodv and hwmp routing protocols in wireless mesh networks.," *2013 IEEE International RF and Microwave Conference (RFM), RF and Microwave Conference (RFM), 2013 IEEE International*, p. 116, 2013.
- [43] "Lpwan-brochure-interactive.pdf." <https://cdn2.hubspot.net/hubfs/427771/LPWAN-Brochure-Interactive.pdf>. (Accessed on 04/25/2017).
- [44] "Lora-faqs.pdf." <http://www.semtech.com/wireless-rf/lora/LoRa-FAQs.pdf>. (Accessed on 04/25/2017).
- [45] "eui64.pdf." <https://standards.ieee.org/develop/regauth/tut/eui64.pdf>. (Accessed on 05/08/2017).
- [46] "packet_forwarder/protocol.txt at master · lora-net/packet_forwarder." https://github.com/Lora-net/packet_forwarder/blob/master/PROTOCOL.TXT. (Accessed on 04/25/2017).
- [47] "lopyspecsheetgraffiti2017newr.pdf." <https://www.pycom.io/wp-content/uploads/2017/01/lopySpecsheetGraffiti2017newR.pdf>. (Accessed on 04/25/2017).
- [48] "Terd-wts-tr-27 - lora_certification_test_report_201701.006.pdf." https://www.lora-alliance.org/portals/0/CertifiedProducts/pycom_lopy/TERD-WTS-TR-27%20-%20LoRa_Certification_Test_Report_201701.006.pdf. (Accessed on 04/25/2017).
- [49] "Micropython - python for microcontrollers." <https://micropython.org/>. (Accessed on 05/05/2017).
- [50] "Lora app server - open-source lorawan application-server." <https://docs.loraserver.io/lora-app-server/>. (Accessed on 05/07/2017).
- [51] "Wirnet station 868 mhz - kerlink | solutions réseaux pour l'internet des objets." <http://www.kerlink.fr/en/products/lora-iot-station-2/wirnet-station-868>. (Accessed on 05/03/2017).

This appendix gives more details about the tests conducted in the thesis.

A.1 Graphical overview

In figure A.1 a graphical overview is shown for each test.

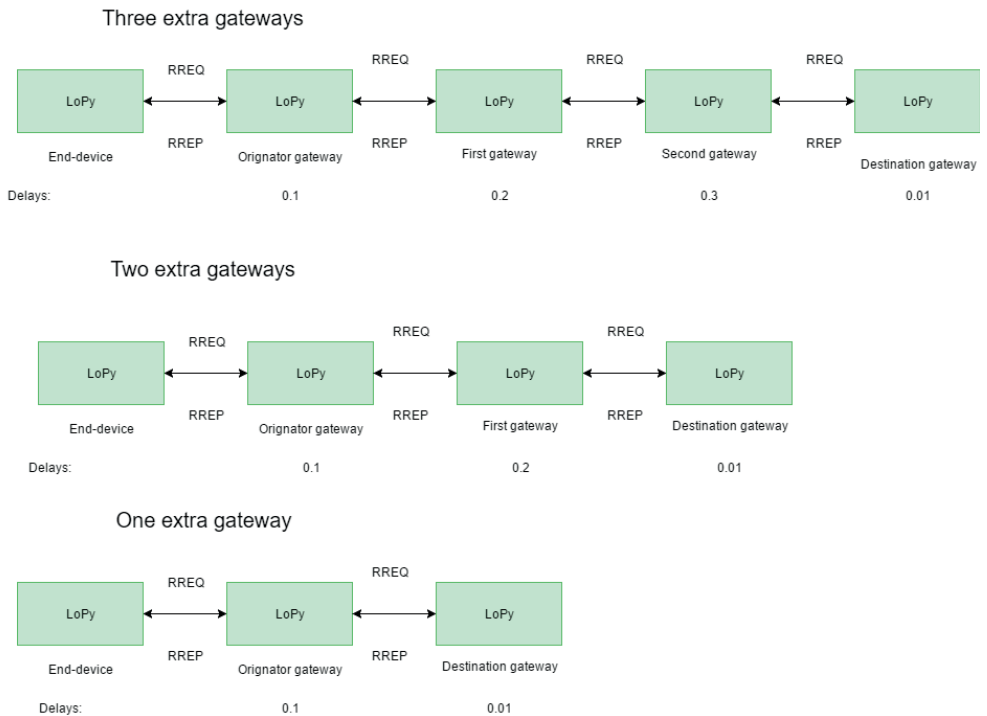


Figure A.1: Graphical overview for tests

A.2 LoPy details

Table A.1: Lopy details

Hardware version	LoPy 1.0
Software version	1.6.12.b1



LUND
UNIVERSITY

Series of Master's theses
Department of Electrical and Information Technology
LU/LTH-EIT 2017-591

<http://www.eit.lth.se>