Abstract

New network protocols are continuously being developed, and a particularly interesting area of research is in ad hoc networks. Due to their dynamic and self-organizing nature with no infrastructure, they introduce properties that are very beneficial in e.g. emergency situations and military applications. However, an additional important property which still remains as a challenging task, is the issue of security and access control.

BATMAN is a new ad hoc network routing protocol which has been modified in order to support identification and authentication of mobile nodes trying to access a restricted network. Routing protocols pose as a critical aspect to performance in mobile wireless networks and an important way of testing and evaluating their design is through simulation.

The goal of this study was to extend the network simulator ns-3 to support simulation of both the original and modified version of BATMAN such that protocol design, interactions, and large scale performance issues could be investigated.

// what did the results show This report shows that the implemented BATMAN protocol does not perform significantly better when compared to DSDV in neither good or lossy networks. However the modified and secure version of the BATMAN protocol.

Preface

This report is written by Anne Gabrielle Bowitz and describes the work done in my master's thesis which is a part of the Master's degree Programme in Communication Technology at the Norwegian University of Science and Technology, NTNU. The was work performed under the supervision of Martin Gilje Jaatun of SINTEF ICT Norway and Stig Frode Mjølsnes from the Department of Telematics at NTNU.

// Experience and thanks!

Abbreviations

AES Advanced Encryption Standard

AODV Ad hoc On-Demand Distance Vector

ATM Asynchronous Transfer Mode

B.A.T.M.A.N Better Approach To Mobile Ad hoc Networking

CBC Cipher Block Chaining

CBR Constant Bit Rate

DARPA Defense Advanced Research Projects Agency

DSDV Destination Sequenced Distance Vector

DSR Dynamic Source Routing

EQ Echo Link Quality

HNA Host Network Announcement

IPv4 Internet Protocol version 4

IV Initial Value

MANET Mobile Ad hoc Network

MSC Message Sequence Chart

ns-2 Network Simulator Version 2

ns-3 Network Simulator 3

OGM Originator Message

OLSR Optimized Link State Routing

PC Proxy Certificate

PC0 Proxy Certificate 0

PC1 Proxy Certificate 1

PDR Packet Delivery Ratio

PKI Public-Key Infrastructure

PKIX Public-Key Infrastructure using X.509

REAL REal And Large

RQ Receiving Link Quality

SP Service Proxy

Tcl Tool Command Language

TCP Transmission Control Protocol

TQ Transmit Link Quality

TTL Time To Live

UDP User Datagram Protocol

VINT Virtual InterNetwork Testbed

Definitions

Ad Hoc Network A self-organizing network with no requirements to pre-existing infrastructure or centralized administration.

Ad hoc On-Demand Distance Vector (AODV) A reactive ad hoc routing protocol [PBRD03].

Advanced Encryption Standard Cipher Block Chaining (AES-CBC) The Advanced Encryption Standard (AES) encryption in Cipher Block Chaining (CBC) mode of operation.

Authentication Fields Two fields added to the Originator Message (OGM) used in the modified BATMAN routing protocol containing extracts or one-time passwords from the Authentication Key Stream.

Authentication Key Stream Key Stream generated with the AES-CBC algorithm taking in a shared symmetric key, an IV value and a nonce as input. It is used to authenticate and tie an a node to the OGMs it broadcasts to its link-local neighbors.

Constant Bit Rate Data traffic generated with a constant bit rate.

Destination Sequenced Distance Vector (DSDV) Proactive A (MANET) routing protocol. Routing updates are broadcasted or multicasted by every node periodically and when there is significant changes in the network topology. Based on the routing updates a node calculates paths to every node in the network using the Bellman-Ford algorithm [PB94].

Discrete Event-based Simulation Simulated network where nodes trigger events, such as a packet being sent, which is stored in a queue sorted by the scheduled event execution time [WvLW09].

Emulator A system that is able to mimic function or behavior of another system.

Empty OGM An OGM used in the modified BATMAN routing protocol containing an empty Authentication field.

Friis Propagation Model Simple transmission formula describing the propagation loss in a traffic flow between nodes [Fri46].

Host Network Announcement (HNA) Message used by a BATMAN Originator to inform other nodes in the network that it can be used as a gateway to another network or host [NALW11].

Initial Authentication Phase Phase where nodes exchange and verifies Proxy Certificate 0 (PC0) before exchanging ephemeral keys and generating authentication key streams.

Interface Address IPv4 address assigned to a BATMAN node which is put in the nodes self-generated OGMs.

MANET Ad hoc network were nodes can be highly mobile.

NIST ATM Network simulator targeted for simulating and analyzing the behavior of Asynchronous Transfer Mode (ATM) networks [GKS95].

ns-2 Discrete event network simulator written in C++ and OTcl based on the REAL network simulator from 1989 [nsnayc].

ns-3 Modular discrete event network simulator written in C++ and Python.

One-Time Password A 16 bit extract from the Authentication Key Stream generated by a node running the modified BATMAN routing protocol.

Optimized Link State Routing (OLSR) A proactive ad hoc routing protocol based on an optimization of the classical link state routing algorithm [CJ10].

Originator Synonym for a node using the BATMAN protocol generating and broadcasting own Originator Messages (OGMs).

Originator List A table maintained by every node in a BATMAN network which consists of information about every other known originator in the network [NALW11].

Originator Message (OGM) A message periodically broadcasted by a originator to inform its link-local neighbors about its presence. [NALW11]

OTcl An object oriented extension of Tcl which is a scripting language.

Packet Delivery Ratio Defined as the ratio between the amount of received packets and the amount of packets actually transmitted.

Proactive Routing Protocol Routing protocols which periodically share routing information in order to maintain their routing tables, also known as table-driven protocol. The opposite of proactive routing protocol is the reactive protocol.

Proxy Certificate A X.509 public-key certificate including a critical certificate information extension used to delegate rights and restrictions within a network [TET$^+$ay].

Public-Key Certificate Electronic document containing most importantly a digital signature which ties a public-key to an identity.

Reactive Routing Protocol Routing protocols that only construct routing paths when they are required, also known as on-demand protocol. The opposite of reactive routing protocol is the proactive protocol.

Restricted Network Network established between nodes using the modified BATMAN routing protocol where one central node acts like a Service Proxy (SP).

Service proxy A central node in a restricted network running the modified BATMAN routing protocol with permission to sign and issue Proxy Certificates (PCs) to other nodes.

SQLite A self-contained and serverless SQL database engine reading and writing directly to ordinary disk files [SQLay].

Symmetric Key A symmetric key exchanged between nodes in a restricted network and used when generating an Authentication Value.

VINT Project Research project funded by Defense Advanced Research Projects Agency (DARPA) which provides improved simulation tools for network researchers to use in the design and deployment of new wide-area Internet protocols [BBE+99].

X.509 Public-Key Certificate Public-key certificate standard managed by the Public-Key Infrastructure using X.509 (PKIX) working group of IETF [HPFS02].

Contents

List of Figures

List of Tables

Chapter 1

Introduction

MANETs have certain characteristics and properties that separate them from traditional computer networks. Most importantly they have no requirements to pre-existing or fixed infrastructure and they need no centralized administration maintaining the network. In such networks, it is the participating nodes' responsibility to sustain routing paths between nodes in the network and make sure that traffic is routed efficiently and reliably from a source to a destination.

Crisis management after a major disaster, emergency situations, and rescue operations are frequently identified as application areas for MANETs [TJÅAN09]. These are situations were resources might be scarce as well as conditions being unpredictable and rapidly changing. Existing infrastructure can not be relied on as it may be damaged or congested and time is restricted. MANETs can be quickly and cost-efficiently deployed, making them very valuable in situation such as these.

The nature of the situations mentioned above imply that providing information security is important in MANETs when they are to be used in such situations. For instance, being able to restrict the access to a network would prevent valuable resources and information being available or wasted on activities not related to the operation. Access control also enables node authentication and confidentiality of information by only allowing authorized nodes to route traffic in the network [TJÅAN09].

Routing protocols are a critical aspect to the functionality and performance in MANETs and pose as a natural place to apply security elements in order to achieve some sort of access control. Several designs of a secure ad hoc routing protocols have been proposed over the years [Neeay] including XX, YY, and ZZ.

BATMAN is an ad hoc routing protocol with a simple and robust routing regime. With it's limited complexity compared to many of it's counterparts/alternatives makes this protocols easier to modify for security purposes. This design is proposed to create a authentication mechanism in a scenario such as the ones mentioned above.

However, with limited resources it is also important that security mechanisms do not substantially affect the overall performance and throughput of the network. Thus the development of such new routing protocol requires testing and evaluation against well-known protocols in various environments. Several network simulators exists providing easily accessible resources to study new protocols and models and has through time been the backbone of MANET research [NCÇ+11].

1.1 Objectives

The objectives of the work done behind this report, is to extend the network simulator, ns-3, to support both the original BATMAN routing protocol as well as the modified and secure version. The protocols' performance are then to be evaluated in a various range of simulation scenarios in the network simulator. Both versions of the BATMAN protocols are also compared against a well-known and already implemented routing protocol in ns-3, namely DSDV

1.2 Limitations

Using network simulation to evaluate the performance of network protocols introduce certain limitations by nature. Network simulators are based on statistical models that attempt to resemble real life scenarios as close as possible. Thus the results retrieved from simulations must only be considered approximations to how a protocols would perform in real life and should only give indications of how their behavior affect their performance. Further discussed in Chapter 4.

Also running simulations is a time and resource demanding task. Thus the amount of various simulations scenarios and simulation runs performed must adjusted with respect to the time and resources available at the time.

1.3 Method

The work behind this project can roughly be split into four parts:

1. Research and study of background material.
2. Implementation of the original BATMAN routing protocol in ns-3.
3. Add support for the security extensions to the BATMAN routing protocol in ns-3.
4. Use ns-3 to conduct simulations in order to observe, investigate and compare the protocol design, behavior and performance of the different versions.

Adding the routing module for the BATMAN protocol in ns-3 was inspired by how the newly added protocol DSDV was implemented as well as the already existing

protocol Ad hoc On-Demand Distance Vector (AODV). These are also protocols that are fairly simple in their functionality compared to other more complicated protocols such as Optimized Link State Routing (OLSR). In addition a lot of time was used to study the real-life implementation of BATMAN in order to get the ns-3 modules to imitate the protocol as closely as possible.

The simulations performed using the BATMAN protocol were based on previous studies such as [NCÇ+11] and [BMJ+98]. The simulations characteristics were also heavily affected by the context in which they are to be used, e.g. emergency situations and military operations.

1.4 Document Structure

The remainder of this report is structured in three main parts. Part one contains necessary background material needed to understand the rest of the report such as explanations of the BATMAN protocol as well as the security elements introduced in the modified version. It also gives an insight into to how network simulation work in addition to descriptions of various popular simulator used in research.

Part two of the report shows how the two BATMAN versions are implemented as routing modules in the ns-3 network simulator.

The last part of the report explains how the simulations have been conducted and the results retrieved from them. Here we also find a discussion of the results and a final conclusion summing up the report.

Some appendices are also included to give a further explanations and deeper understanding of certain areas. It is remarked in the text if the subject is further described in an appendix.

Part I

Background

Chapter 2

BATMAN - Ad Hoc Routing Protocol

This chapter provides the necessary background material required to understand the basic dynamics behind ad hoc routing and the workings and behavior of the BATMAN routing protocol.

2.1 Ad Hoc Network Routing

In an ad hoc network, it is the participating nodes' duty to maintain and control the communication within the network. This entails that nodes not only send and receive data to and from each other, they must also relay traffic on behalf of other nodes like a router.

How the traffic is routed through the network depends on the nodes' routing tables which are maintained by a routing protocol.

Nodes in ad hoc networks are characterized by having the ability to be mobile and still being able to route traffic in the network as long as they are within transmission range. Networks where the nodes are mobile are often referred to as As (MANETs).

The mobility of the nodes and the lack of infrastructure create a very dynamic network with a topology that changes randomly and frequently. Due to this unique behavior, regular network functionality such as routing is a challenging task. Classical routing protocols are not well suited for such networks, thus a number of specialized protocols have been developed over the years. The most prominent routing protocols for ad hoc networks are arguably AODV, DSDV, Dynamic Source Routing (DSR), and OLSR [NCÇ+11].

Several new protocols for ad hoc networks are continuously being designed and amongst these we find the relatively new protocol called BATMAN which is an abbreviation for "Better Approach To Mobile Ad hoc Networking". The sections below describe this protocol in further detail.

2.2 The BATMAN Routing Protocol

The development of the ad hoc routing protocol BATMAN was first initiated by a group of developers working on the OLSR protocol. They felt OLSR contained significant shortcomings and that the changes made to it in order to fix them were breaking compatibility with the original protocol as described in RFC3626 [CJ10]. Thus, a group of developers decided to design a new and simpler routing protocol called BATMAN that could hopefully become a better alternative to OLSR [Mesayb].

The principle of the BATMAN routing protocol is simple; nodes, or Originators as they are also referred to, build their routing tables by flooding the network with routing updates called BATMAN packets. The strategy for a node when finding the best routes in the network, is to determine for each destination one single-hop neighbor which can be utilized as the best next-hop towards the destination [NALW11].

The packet flooding starts with the Originators in the network periodically broadcasting BATMAN packets thus informing their link-local neighbors about their existence (Step 1). Neighbors receiving the packets will rebroadcast them to their own link-local neighbors (Step 2) which will in turn rebroadcast them again to their neighbors (Step 3) and so on. The packets originated from the first node are in this way eventually flooded through the entire network as illustrated in Figure 2.1.

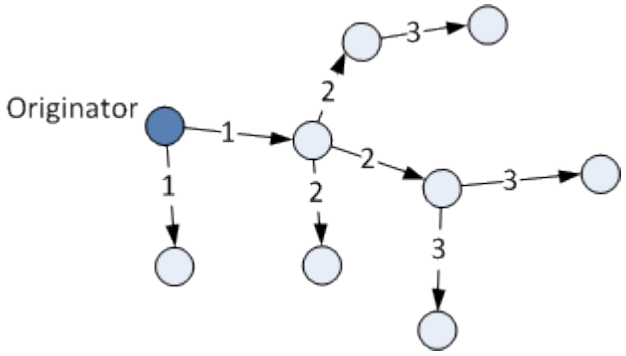


Figure 2.1: Illustration of how a BATMAN packet is flooded through a network.

A BATMAN packet is flooded through the network until every node has received it at least once, or until the packet's Time To Live (TTL) has expired, or until it is lost due to lossy communication links. The details of a BATMAN packet are found in Section 2.2.1.

The BATMAN routing protocol has over the years gone through several implementation and testing phases where improvements and changes have been made. The core algorithm has evolved, and at the time of writing the algorithm is at generation IV [Neeay].

The latest generation added three new fields to the BATMAN packet, namely Pre-

vious Sender, TQ and HNA Length. These fields where added to improve the protocol's handling of asymmetric links, reduce routing overhead, and enable packet aggregation.

For simplicity the rest of this chapter first explains the BATMAN algorithm III. The last section of the chapter is devoted to describing generation IV which basically has just added functionality on top of the previous algorithm.

2.2.1 Packet Formats

A BATMAN packet consists of one Originator Message (OGM) together with zero or more attached Host Network Announcement (HNA) extension messages. The HNA messages are used when an originator wants to announce that it is connected to another network or host.

However, this feature added by the HNAs is not a critical factor when understanding how the basic BATMAN routing algorithm works. This report focuses on how the performance of this basic routing is affected by adding some security extensions explained in Chapter 3. Thus, the HNA messages will not be implemented in the network simulator and therefore not described any further here. However, some more information about these messages can be found in Appendix A.1. Even though the HNA messages is omitted, we sometimes refer to OGM as a BATMAN packet.

The OGM contains the most important information used by the BATMAN routing algorithm. These messages have a fixed size of 12 Bytes and an illustration of their general format is shown in Figure 2.2.

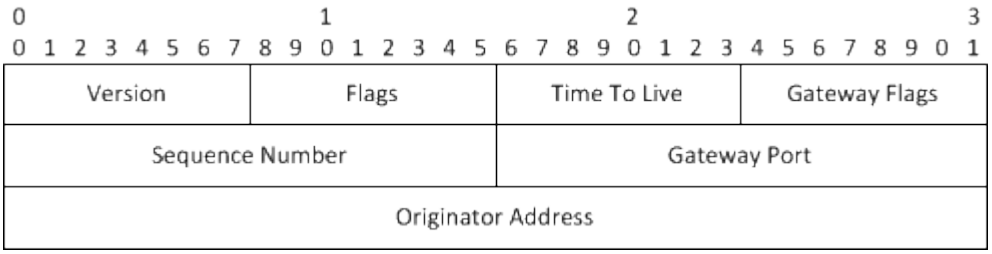


Figure 2.2: Originator Message (OGM) Format.

The different fields in the OGM are shortly explained in the list below:

- **Version**
  Identifies the BATMAN version of the contained message.
- **Flags**
  Bits indicating whether a node is a link-local(direct) neighbor or not.
- **Time To Live (TTL)**
  Is used to limit the number of hops an OGM can do in the network.
- **Gateway Flags**
  Indicates if the node can act as a gateway with access to the Inter-

net.

- **Sequence Number**
  Number added by an originator to every OGM it generates. The number is incremented every time a new OGM is generated.

- **Originator Address**
  The Internet Protocol version 4 (IPv4) address belonging to the originator on which behalf the OGM was generated.

The Gateway Flags are used when a node wants to announce that it can act like a gateway to the Internet. These flags are also not utilized when simulating and is thus not described further. How the remaining fields are used by the BATMAN routing algorithm, is explained in the following sections.

2.2.2 Originator List

Every node running the BATMAN routing protocol maintains information about all other known originators in a list called the Originator List. The list contains one entry for each originator from which it has received an OGM and is used by the node to choose the best next-hop to a destination.

For each known originator, the most important information a node must maintain in its Originator List, is the following [NALW11]:

- **Originator Address**
  The IPv4 address of the originator as given in the corresponding field of the received OGM.

- **Last Aware time**
  A timestamp that is updated whenever an OGM is received from the given originator.

- **Current Sequence Number**
  This field holds the sequence number from the last accepted OGM from the given originator. This is described in Section 2.2.3.

- **HNA List**
  All announced networks of the originator including their IP-range and netmask.

- **Gateway Capabilities**
  If the originator offers a gateway, this field contains its announced parameters and relevant information.

- **Neighbor Information List**
  For each link-local neighbor of the node from which it has received an OGM, the following information must be maintained:

  - Sliding Window:
    For each In-Window sequence number it is remarked if an OGM with this sequence number has been received. Further explained in Section 2.2.3
  - Packet count:
    The amount of sequence numbers recorded in the Sliding Window. This value is used as a metric when choosing the best next hop to the originator.
  - Last Valid Time:

The timestamp when the last valid OGM was received via this neighbor.

– Last TTL:
The TTL of the last OGM which was received via this neighbor.

This list functions as a node's routing table and is used when routing traffic in the network.

// more about the neighbor information list

How paths, or best next-hops, to other nodes are found using this information, is explained in next section.

2.2.3 Sliding Window and Neighbor Ranking

The sequence number in a received OGM is the key information used by the routing algorithm when choosing the best next-hop towards a destination. Here it is the amount of accepted sequence numbers recorded from a neighbor which is used as a metric for the quality of links.

BATMAN utilizes a Sliding Window to keep track of the most recently received sequence numbers. For a sequence number to be accepted it must reside within the sliding window which is a set of numbers that slides across the total range of numbers set aside for sequence numbers. More about how the Sliding Window works, is described in Appendix A.2.

In principle this means that a node chooses the best next-hop towards a destination depending on which link-local neighbor it has received the highest amount of accepted OGMs that originated from the destination.

Figure 2.3 illustrates this with node A which has two neighbors, S and T, that can be used as a next-hop towards node B. Node A continuously receives OGMs which originates from node B but are rebroadcasted from both node S and T. Node A ranks its neighbors by calculating how many rebroadcasted OGMs (sequence numbers) from node B it has received via each neighbor.

Originator	Neighbors	PacketCount
B	S	27
	T	41

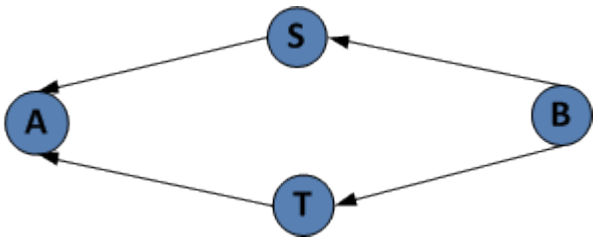


Figure 2.3: An illustration showing the flow of rebroadcasts from node B to node A and a very simplified version of node A's Originator List.

The BATMAN protocol also ensures that the best single-hop neighbor is only selected if it is reachable by a bidirectional link. This is further explained in the next section.

2.2.4 Maintaining the Originator List

Maintaining the originator list involves broadcasting, receiving and rebroadcasting OGMs. To inform other nodes about its presence, an originator generates and broadcasts its own OGMs periodically with its own interface address in the Originator Address field.

Upon reception of an OGM, a node will silently drop the message before further processing depending on some simple rules. Some of these are:

- Version stated in the OGM's version field does not match the nodes own internal version.
- Sender address of the packet is the node's own interface address. This means that it's own broadcast has just been echoed back to the node.

If however a node receives an OGM that contains its own interface address in the Originator Address field, it means that the message was originally generated and broadcasted from this node. As illustrated in Figure 2.4, an OGM has been rebroadcasted from neighbor B back to the originator A. This is an indication that the link can be used in both directions, also called a bidirectional link.

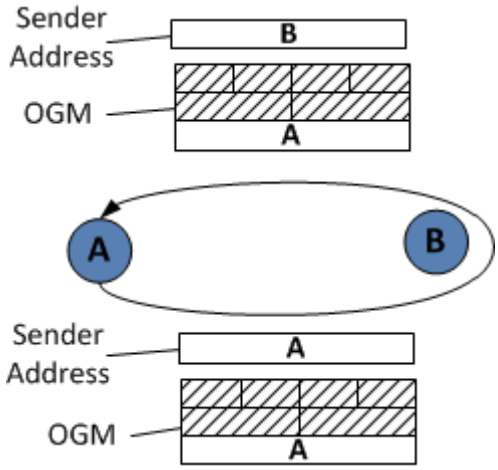


Figure 2.4: Illustration of node A receiving a rebroadcast of its own OGM from node B.

Before the OGM is rebroadcasted by node B, some fields must be changed such as the TTL field and the Direct-link flag. All other fields are left unchanged.

If a received OGM is not dropped or a self-generated OGM, a node will rebroadcast

it after processing if it was received from a bidirectional link. A node considers a link to a neighbor to be bidirectional if it has received a self-generated OGM with the Direct-link flag set from that neighbor within a reasonable time.

Lastly, if a node has not received any OGMs from an Originator in its Originator List for a time longer than some timeout interval, this entry in the Originator List is removed.

2.3 BATMAN IV

Even though the BATMAN routing algorithm III ensures that only bidirectional links are used when routing packets, the protocol showed significant shortcomings when handling asymmetric links [Mes11]. To improve this weakness the developers decided to add an 8 bit Transmit Link Quality (TQ) field which is explained in Section 2.3.1.

By periodically flooding OGMs, the BATMAN protocol creates a significant amount of overhead in networks that are dense and without heavy packet loss [Mes11]. To reduce the routing overhead the developers introduced the Previous Sender field which ensures that a node doesn't rebroadcast the same OGM more than once. More about this in Section 2.3.2

The third and last field which was included in the OGM, is the HNA-lenght. This was added to enable packet aggregation which combines several distinct OGMs into one packet before broadasting it.

The new OGM format after adding the three new fields is shown in Figure 2.5.

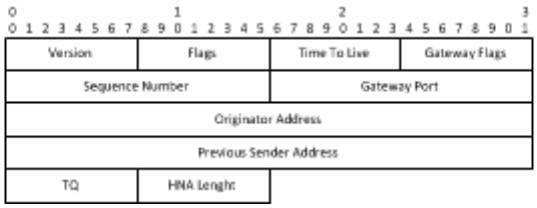


Figure 2.5: Format of the OGM in BATMAN IV.

2.3.1 Transmit Link Quality (TQ)

Link quality in BATMAN IV is divided into two parts, Receiving Link Quality (RQ) and Transmit Link Quality (TQ). The RQ value is the amount of packets a node receives from its neighbors.

BATMAN also keeps track of Echo Link Quality (EQ) which is the number of

rebroadcasts of a node's own OGMs received from its direct neighbors as illustrated in Figure 2.6.

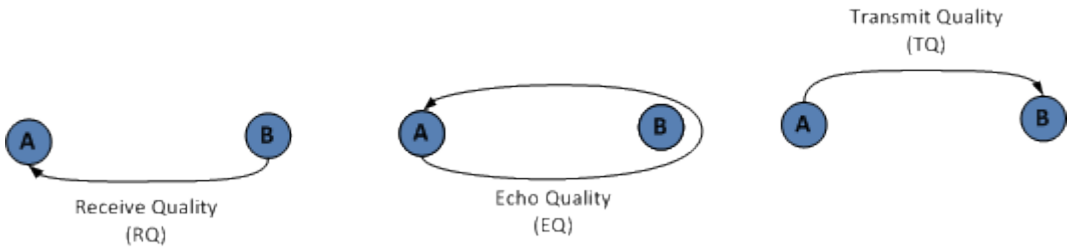


Figure 2.6: Illustration of RQ, EQ and TQ.

Using the values RQ and EQ, a node can derive the local TQ towards a direct neighbor by performing the following calculation:

$$EQ = RQ \cdot TQ_{local} \Rightarrow TQ_{local} = \frac{EQ}{RQ}$$

This local link quality is propagated through the network to inform other nodes about the transmit quality.

When a node generates own OGMs, the TQ field is set to the maximum lenght (255) before it is broadcasted. A receving direct neighbor calculates its own local link quality and adds it to the received TQ value before rebraodcasting the OGM with the new TQ value.

The new TQ value is found by performing the following calculation:

$$TQ = TQ_{received} \cdot TQ_{local}$$

An exampel of how the TQ values are calculated is shown in Figure 2.7.

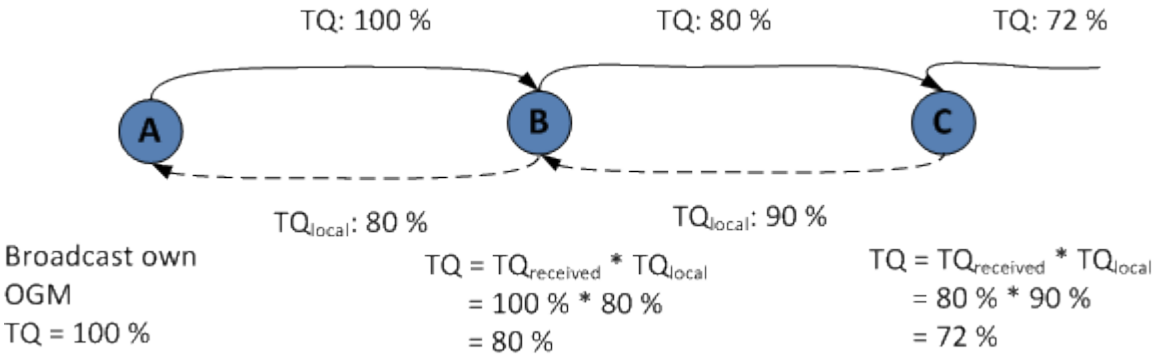


Figure 2.7: Example of how the TQ is calculated between three nodes.

2.3.2 Previous Sender

The Previous Sender field was added to reduce echo rebroadcasting which produced unnecessary overhead in the network. This echo cancellation ensures that a node rebroadcasts the same OGM only once after it has received it.

Upon reception of an OGM a node changes the Previous Sender field to the address of the node it received it from before rebroadcasting it. Thus, if a node receives an OGM with its own address in the Previous Sender field, it will silently drop the message.

This echo cancellation is illustrated by an example in Figure 2.8.

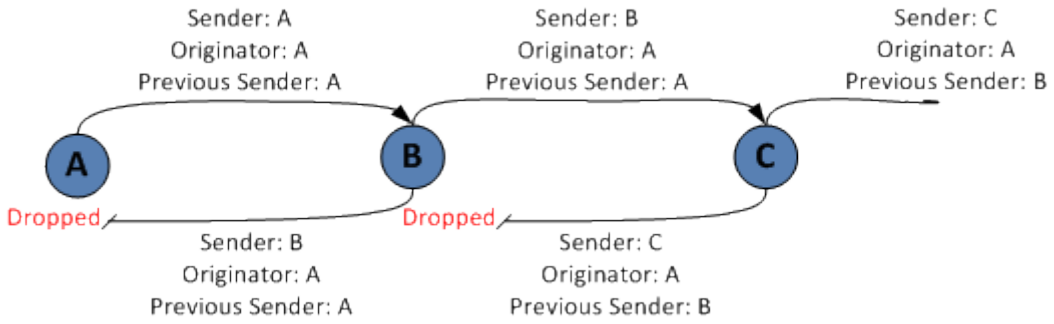


Figure 2.8: Illustration of echo cancellation.

2.3.3 Asymmetric Link Handling and Hop Penalty

To help ensure that the best bidirectional links are chosen by the BATMAN protocol, an additional value was introduced to penalize links that have poor Receiving Link Quality (RQ).

This penalty is a weighted value and is found with the following function:

$$f_{asym} = (100\% - (100\% - RQ)^3)$$

The penalty has a big influence on the TQ value for links with large packet loss, and only a small influence on links with little packet loss [Mes11].

A node using the BATMAN protocol is only aware of the best next hop towards a destination and not the entire route. Thus, the node does not know how many hops the route may consist of. In some networks it might be desirable to choose the shortest path, as in the minimal amount of hops, to a destination in order to reduce latency and save bandwidth [Mes11]. Therefore a hop penalty was also introduced where every hop an OGM does decreases the TQ value by a fixed amount.

Both the asymmetric penalty and the hop penalty are added to a TQ value from a

received OGM message. The final calculation a node must perform on the TQ value before rebroadcasting the OGM, is as follows:

$$TQ = TQ_{received} \cdot TQ_{local} \cdot f_{asym} \cdot hop\_penalty$$

More detailed information about how the BATMAN ad hoc routing protocol works, can be found in Appendix A. The next chapter explains how this BATMAN routing protocol is modified in order to include features such as node identification and authentication.

Chapter 3

BATMAN Security Extensions

Now that we have an understanding of how the BATMAN protocol works, it is time to investigate where we can introduce security elements, modifications, and extensions to provide authentication mechanisms and access control in an ad hoc network. This chapter first covers some of the common security issues in ad hoc networks and then it continues to describe how the BATMAN protocol is modified.

The security design is based on a previous project assignment done at NTNU [BG10].

3.1 Security in Ad Hoc Networks

Several challenges and issues are prominent due to ad hoc networks' unique characteristics as discussed in Section 2.1. The issues affect the networks' design, deployment, and performance in addition to how security should be implemented. Amongst the major issues relevant for this thesis, we find:

- Resource constraints
- Unreliable medium
- Mobility of nodes

Mobile nodes used in wireless networks usually have limited resources available restricting their computing capacity and battery power [YLY+04]. This affects the choice of cryptography-based security mechanisms which might be computation-intensive to perform.

Also, communication is done on a shared radio channel making it easy for a malicious node to eavesdrop and perform attacks on the network. The communication medium is also very unreliable compared to wired networks, which might result in high packet loss ratio [NALW11].

Because of ad hoc networks non-hierarchical topology, classical and common security measures are not well suited. Traditional computer networks are infrastructure-based with central entities, such as routers and gateways, which create natural points

in the network to add security elements.

Finally, due to the mobility of the nodes, they can frequently join and leave a network at any point in time. If there is no authentication mechanism present, there is no association or relationship between nodes or networks making it easy for an intruder to join a network and carry out an attack.

3.2 Areas of Application

MANETs have a vast range of application areas. The list below mentions some of the most common situations mentioned in research papers were MANETs are applicable:

- military operations
- crisis management after a major incident (war or natural disaster)
- emergency and rescue operations
- wireless sensor network
- collaborative and distributed computing

The protocol's design proposed and described in this chapter mainly focuses on being utilized in situations such as emergency and rescue operations. It is important to consider the area of use for the network in order to consider the possible influences this (the scenario) might have on how security should be applied to the protocol.

In an emergency or rescue situation, it is natural to have groups/collection of actors with different roles and duties which are in charge of various tasks during the operation. Usually there is always one actor or group of actors which are in charge of the administration and management of the entire operation [Dir07], such as the police during a search and rescue operation. Actors who want to participate in the operation normally have to report to this central administration in order to be assigned a role or delegated a duty [Dir07, Neeay].

3.3 General Concept

The overall goal of the modified BATMAN protocol is to make nodes which are a part of a restricted network only accept routing updates (OGMs) from other nodes that are appropriately authenticated. This entails that they must be in possession of some authentication token which proves their identity.

By only accepting OGMs from nodes that are authenticated, all other traffic sent in the network will only be routed to or via trusted nodes. How this is accomplished is explained in the following sections.

3.4 Proxy Certificates (PCs)

The tokens used to authenticate nodes are Proxy Certificates (PCs) [TET+ay]. These certificates are special versions of traditional X.509 Public-Key Certificates containing a critical certificate information extension.

The extension indicates that the certificate is a Proxy Certificate (PC) and it contains fields such as Proxy Path Length Constraint, Proxy Policy, and Proxy Certificate Path [TET+ay].

The policy field is the part of PCs that makes them beneficial to use as an authentication token in this context. This field enables the possibility of finer granularity of access control by defining rights and restrictions in the network on the node which it is issued to.

Thus, a node may not only be authenticated with its PC, but can also be delegated different restrictions and rights in the network on behalf of the issuer.

3.5 Service Proxy (SP)

The node or entity who signs and issues PCs must have a central role with a higher level of trust than other participating nodes in the network.

Introducing a central entity in a network which is characterized as being infrastructureless, might seem less than ideal. However, as mentioned in Section 3.2 the area of use for this kind of ad hoc network, it is common to find a central actor who manages and administrates situations such as an emergency search and rescue operation [Dir07].

Therefore, it is natural to assign this responsibility of signing and issuing PCs to a central node in the network. We refer to this node as a Service Proxy (SP) and it is in possession of a self-signed PC, called Proxy Certificate 0 (PC0), which it uses to sign other PCs which will be referred to as Proxy Certificate 1s (PC1s).

3.6 Continuous Authentication and Broadcasting of OGMs

As mentioned in Section 3.3, nodes may only accept and process OGMs from other nodes which have been properly authenticated. This means that every OGM broadcasted in the network must somehow prove that it has been sent by an authenticated node and that it has not been altered in transit. This can be accomplished by digitally signing the OGM with the node's private key from its PC0 [RSA78]. Confidentiality of the OGM can also be achieved by encrypting the message with the receiver's public-key.

However, every participating node in a BATMAN network by default generates and broadcasts OGMs every second. In addition they also rebroadcast received OGMs in between their own OGMs, creating a lot of traffic in the network. Thus, signing and encrypting every OGM a node transmits as well as validating every message received, would be computationally infeasible given the nodes' restricted resources as discussed in Section 3.1.

To solve this issue of tying a node's identity to it's OGMs and validating received OGMs without introducing a significant amount of work, the following solution is proposed and explained in the next sections.

3.6.1 Authentication Key Stream

A node generates a symmetric key, referred to as the ephemeral key K, which it unicasts together with a nonce n and a initial value IV to all of its authenticated link-local neighbors[1]. The message is digitally signed for integrity and encrypted with the neighbor's public key from their PC1s for confidentiality before it is transmitted to the neighbors as shown in Figure 3.1.

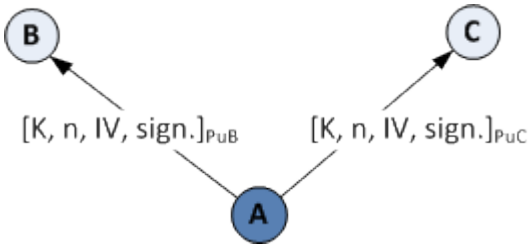


Figure 3.1: An example of node A encrypting the Symmetric Key K with its direct neighbors public-keys PU_B and PU_C, and transmitting it to them.

After the transmission, node A and its direct neighbors use the values from the message to generate a key stream using AES-CBC encryption repeatedly. AES-CBC is the Advanced Encryption Standard (AES) encryption algorithm in Cipher Block Chaining (CBC) mode of operation. This means that a block of plaintext is XORed with the previous ciphertext block before it is encrypted with AES [FKGay] as illustrated in Figure 3.2.
In this case it is the nonce value which is used as the plaintext to be encrypted. In order to generate a large key stream based on the values received, the AES-CBC encryption is repeated where the same nonce value is used. To assure that the ciphertext generated is independent from the plaintext, the nonce value is changed for each repetition. The point behind making the key stream as large as reasonably possible, is due to the nodes' frequent broadcasts. If the key stream is not large enough, this value will be exhausted quickly and more time and resource demanding computations is required.

[1]Entails that it has already received and verified their PC1s which is done in the initial authentication phase explained in Section 3.7.1

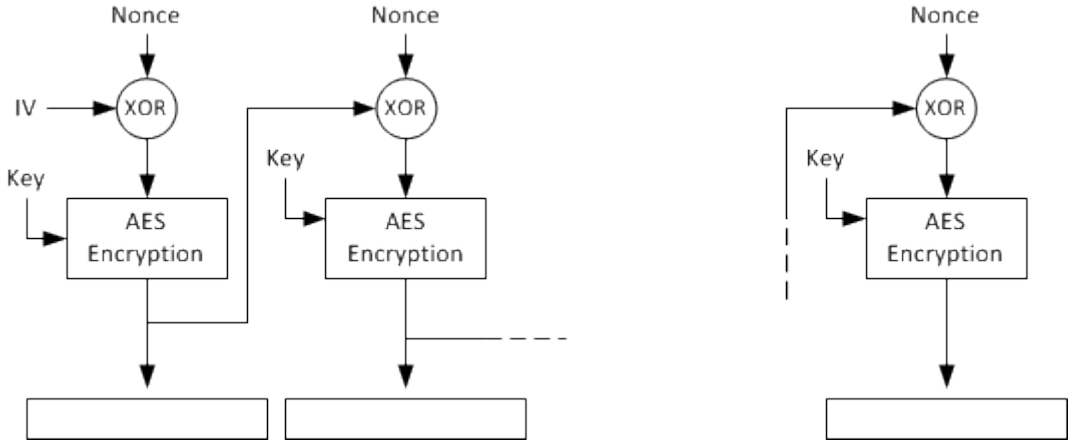
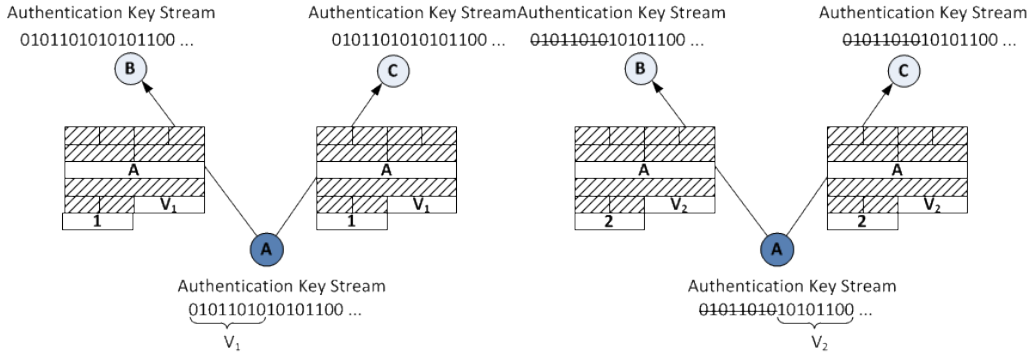


Figure 3.2: An illustration of AES-CBC encryption.

In this way, using the nonce together with the Initial Value (IV) and ephemeral key K as input, the AES-CBC encryption creates a chain of ciphertext blocks which is referred to as an Authentication Key Stream.

Since all the nodes A, B and C know the same input values, nonce repetition rules, and key stream algorithm, this authentication key stream will be the same for everyone.

As illustrated in Figure 3.3, every OGM which is broadcasted by node A from then on, contains an 16 bit extract of this authentication key stream called a One-Time Password. It also appends a 16 bit sequence number that indicates which part of the stream the one-time password is taken from.



(a) OGMs containing value V_1 and Offset 1.(b) OGMs containing value V_2 and Offset 2.

Figure 3.3: Illustration of node A's OGM transmissions containing extracts of the Authentication Value V.

Upon receiving an OGM, the neighbors B and C verify the one-time password by comparing it to the corresponding Authentication Key Stream they generated them-

selves.

The neighbors B and C also create their own Ephemeral Keys, nonces and IVs and transmits them to their direct neighbors just as node A. This leads to every node in the network being in possession of their own Authentication Key Streams in addition to one key stream for every direct neighbor.

The point of a node generating a key stream from the values received from a neighbor, is to be able to verify that future OGMs is actually sent from this specific neighbor.

Authentication is done hop-by-hop which means that a node only authenticates its link-local neighbors even if an OGM did not originate from them. So if node B or C are to rebroadcast an OGM received and originated from node A, they replace the one-time password put there by A with their own one-time password from their own Authentication Key Streams. This creates a from of "web of trust" where a node trusts the originator of an OGM if it is trusted by one of it's direct neighbors.

//maybe an illustration of this as well.

After a certain time interval, nodes will generate new Ephemeral Keys K and repeat the behavior as explained above. This is to sustain a continuous authentication of the nodes.

To make room for the one-time password and sequence number appended to every OGM sent, two new fields is added as shown in Figure 3.4.

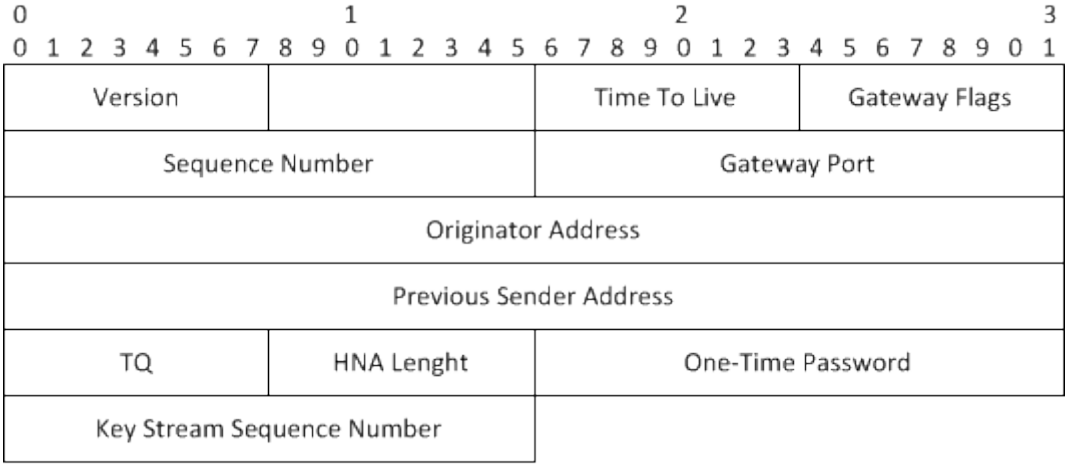


Figure 3.4: Illustration of the OGM format including the fields reserved for security elements.

For simplicity, let's refer to both these fields under the term Authentication Fields. If nothing else it specified, when using the abbreviation OGM from now on refers to this modified version of the original OGM.

3.7 PC Signing and Issuing

A node who wants to join a restricted network must be in transmission range of the network's responsible Service Proxy (SP) in order to be issued a PC1. The idea behind this design choice is that this authentication step should be done out-of-band where the person operating/in charge of/in possession/with the role as the SP/operating as SP will authenticate the the person operating the node face-to-face. XXX

When not part of any restricted network, a node broadcasts its OGMs as usual, but however with empty authentication fields. We refer to this as an empty OGM.

Upon receiving an empty OGM, the SP will engage in a certificate issuing process with the node. This process is similar to how a regular X.509 certificate is issued in traditional networks [Neeay].

First the SP who received an empty OGM sends an invitation to the originator asking it to create and return an unsigned Proxy Certificate (PC). The originator generates a public-private key pair and sends the public-key in an unsigned PC back to the SP. The SP then signs the certificate with its own PC0 before returning it to the originator. This process in presented as a Message Sequence Chart (MSC) in Figure 3.5.

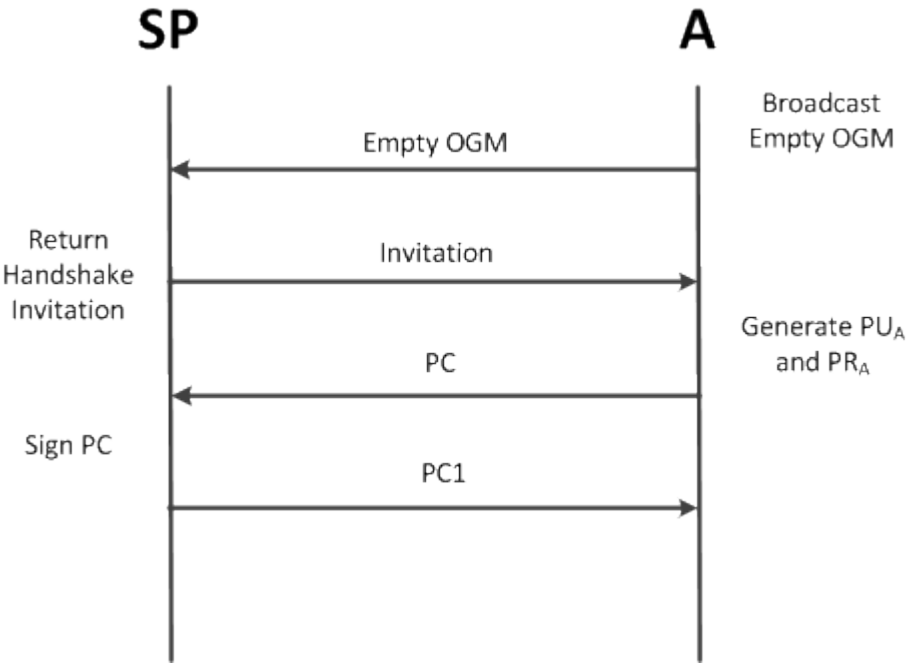


Figure 3.5: Illustration of the OGM format including the fields reserved for security elements.

The originator node is now a part of the restricted network and can be authenticated by all other nodes possessing a PC1 signed by the same SP. How this verification of each others PCs between nodes is performed, is explained in the next section.

3.7.1 Initial Authentication Phase

Before nodes can exchange ephemeral keys and generate Authentication Key Streams, they need to exchange and verify PC1s. This step is called the Initial Authentication Phase.

The phase is triggered if a regular node receives an OGM from a neighbor it has not yet authenticated. The nodes then continue to exchange their PC1s and the certificate verification process follows the standard X.509 procedure [Neeay].

If the certificates are valid, the nodes exchange ephemeral keys and generate authentication key streams and continue as explained in Section 3.6.1.

If the verification process fails, the nodes will not consider each other as authenticated nodes, thus dropping every OGM sent from them. A node also drops an OGM if the one-time password in the message is not verified.

If a node has verified a neighbors PC, but has not yet received a ephemeral key from it, the node will drop the OGMs until it eventually receives a the necessary information needed from which it can generate an authentication key stream.

3.8 Network Entities

The entities and elements introduced to the BATMAN protocol are summarized and shortly described in Table 3.1.
The nodes also need to keep track of their authenticated direct neighbors including their public-key, ephemeral key K, authentication key stream and last received key stream sequence number. This information is added to the Originator List which is maintained by every node as described in Section 2.2.2.

Entity	Description
Service Proxy (SP)	Allowed to sign Proxy Certificates which will be issued to nodes after they have been through an authentication process with the SP.
Proxy Certificate (PC)	Certificate generated by a node that has not been signed yet.
Proxy Certificate 0 (PC0)	Self-signed certificate belonging to a SP. This PC will have the certificate depth of 0 thus we refer to it as a PC0.
Proxy Certificate 1 (PC1)	Certificate that can only be signed by a SP. A node in possession of a PC1 is fully trusted node in the network managed by the SP who signed the certificate.
Ephemeral Key K	A symmetric key generated by every node in the network and unicasted together with a nonce and IV value to every direct neighbor.
Authentication Key Stream	Key stream generated from the AES-CBC algorithm using the ephemeral key, IV and nonce as an input.

Table 3.1: New entities and elements introduced to the modified BATMAN protocol.

Chapter 4

Network Simulation

To get an understanding of the importance of simulating new network protocols, this chapter first explains some of the basic knowledge about network simulation as well as limitations and advantages this tool might introduce. The chapter then covers some of the most popular network simulators used in research and education.

4.1 Network Simulation

New network protocols are continuously being designed and developed with the goal of optimizing various operational requirements such as security, reliability, network scaling, mobile networking, and quality-of-service support [BEF+02]. Studying protocols, both individually and interactively, under varied conditions is critical in order to understand their behavior and characteristics.

Building testbeds and labs to test and evaluate a network protocol can however be both difficult and expensive, especially in large-scale environments. In addition, testbeds and labs are not always capable of reproducing some networking phenomenas, such as wireless radio interference, thus making it difficult to compare and evaluate protocol designs. Lastly they can also be difficult to reconfigure and they have limited flexibility [BBE+99].

Due to the challenges of making real-life models of communication networks, large-scale network simulation has become an increasingly important tool to evaluate protocol design. Various network simulators provide a rich opportunity for efficient experimentation and provide various, but controlled and reproducible network conditions.

Even-though network simulators have several advantages compared to testbeds and labs, they also have their limitations and drawbacks. Some of these are discussed in the next section.

4.1.1 Limitations

The goal of simulations is to model real-life systems as closely as possible. To imitate real-life network phenomenas, e.g. propagation loss or node mobility, different mathematical and statistical models need to be created that describe the occurrence and behavior of different events and processes.

However, such models can never perfectly resemble the unpredictable behavior of the real world. The goal is to have an approximation which produces results that are good estimates of how it would have been in the real world [Neeay].

The details of a simulation is dependent on computer resources and power. Computer limitations, such as memory and processing time, can for instance limit the number of network objects (nodes, links, and protocol agents) that a designer can simulate [BEF+02].

All simulators adopt some level of abstraction which means that they can have a configurable level of detail for different simulations. Users are with this able to trade simulator performance against level of detail by adjusting the level of simulator abstraction. However, this introduces a risk of decreasing simulation accuracy when increasing the level of abstraction [BBE+99].

4.1.2 Simulation Scenarios

As mentioned above, different mathematical models define the behavior of a simulation. By applying sets of initial parameters and other variables, the behavior can be changed to attempt to generate a representative scenario.

When generating simulation scenarios the following requirements should be assessed [BEF+02]:

- network topologies that define links and their characteristics
- traffic models that specify sender and receiver locations and demands
- network dynamics that include node mobility in addition to node and link failure

Some of the conditions that need to be defined include for instance transmission range, representative data traffic models, and realistic models of the movement of the mobile nodes.

The models, conditions and other simulation specifications describing the scenarios in which the BATMAN routing protocol was simulated, is covered in Chapter 6.

4.2 Network Simulators

There are several network simulators with varying focuses targeting different areas of research. Some only focus on a particular network type or protocol such as the NIST ATM simulator, while others including ns-2, ns-3, REAL, Opnet and Insane, target a wider range of protocols [BEF+02].

However, using only one common simulation environment across research efforts can yield many substantial benefits, including [BBE+99]:

- improved validation of the behavior of existing protocols
- a rich infrastructure for developing new protocols
- the opportunity to study large-scale protocol interaction in a controlled environment
- easier comparison of results

Over the past decade, several network researchers have preferred using the network simulator ns-2 when evaluating network systems and protocols [HRFR06]. However, on the basis of some of ns-2's shortcomings and issues, a new simulator, ns-3, has been developed over the recent years. Both simulators are described in the sections below.

4.2.1 ns-2

The Network Simulator (ns) 2, ns-2, is a simulation tool primarily targeted for networking research and educational use. The simulator derives from the old network simulator REal And Large (REAL) from 1989 which was developed with the motivation of studying flow and congestion control schemes in packet-switched data networks [BBE+99, Kes11].

In 1995 the first generation of ns was completed through the VINT project with the hope of becoming a common simulator with advanced features to change the then prominent protocol engineering practices [BEF+02]. The simulator continued to evolve and the second generation, ns-2, was first released in 1996 and was a major architectural change from ns-1 because of its split-level programming model explained below [HRFR06].

ns-2 uses a discrete-event processor as its engine. The simulator provides as mentioned a split-level programming model where the simulation kernel is written in C++ and the simulation setup is done in OTcl, an object oriented version of the scripting language Tcl. The simulation kernel in C++ is responsible for the core set of high-performance simulation primitives while the simulations setup is responsible for the definition, configuration, and control of the actual simulation scenarios [BEF+02].

By utilizing both C++ and OTcl, the simulation maintenance, extension and debugging was separated from the actual simulation itself making it easier to use. In addition the developers and researchers avoided having to recompile the simulator every time a structural change was made which significantly reduced the total amount of recompilations which were time-consuming in the timeframe the simulator was introduced [HRFR06].

Being an open source project, the simulator has substantial contributions from researchers and people who have used the simulator. However, ns-2 is currently only lightly maintained due to the development work on ns-3 which is explained in the next section.

4.2.2 ns-3

As mentioned in Section 4.2, the ns-2 suffered from several issues which prevented the simulators scalability, extensibility and usability. Some of the key issues include:

- **Split-level programming model**
  Few people are familiar to the C++/OTcl structure and the C++/OTcl linking is poorly documented making it a hard to learn and debug. It also puts restrictions on how objects may be combined in new ways [HRFR06].

- **Scalability**
  This important property is considered one of the major concerns about ns-2 cited by its users [HRFR06]. Because of its sequential execution with a single event processing loop running on a single processor, this might become a bottleneck when attempting to simulate more sophisticated communication models e.g. wireless or higher-rate links on a single machine [Neeay].

- **Realism**
  ns-2 packets are not serialized and deserialized making them unable to function with real-world systems [Neeay].

- **Integration**
  ns-2 does not offer many opportunities to integrate with external software, e.g. traffic generators such as iperf and tcplib, or network analysis software such as Wireshark and tcptrace [HRFRay].

In 2006 the development of a new version of ns-2 was initiated. The main project goals were to develop a simulator that was modular and easily extensible, put more focus on collection of data and simulations statistics, had attention to realism and software integration as well as being flexible. In addition, there was more focus on maintaining an extensive documentation, API [HRFR06].

ns-3 is also a discrete-event network simulator written purely in C++ with an optional Python interface. Simulation scripts describing simulation scenarios can be written in either C++ or Python.

ns-3 mirrors real network components, protocols and APIs. For instance, a network packet simulated in ns-3 is implemented in detail and is serialized and deserialized when transmitted and received respectively.

This increased realism makes the code more reusable, modular, and portable [Neeay]. It also enables the simulator to be used as an emulator in environments including real hardware, software, and networks [Neeay].

In order to avoid the issues which ns-2 suffered from discussed above, ns-3 had to break compatibility with it's predecessor. This entailed that the relatively large amount of supported protocols in ns-2 could not be directly ported to ns-3, but had and still has to be re-implemented.

Since the simulator is still fairly new, few protocols and other network applications and functionality are yet to be supported by the ns-3. Figure 4.1 shows the different modules that are currently supported by the ns-3 simulator:

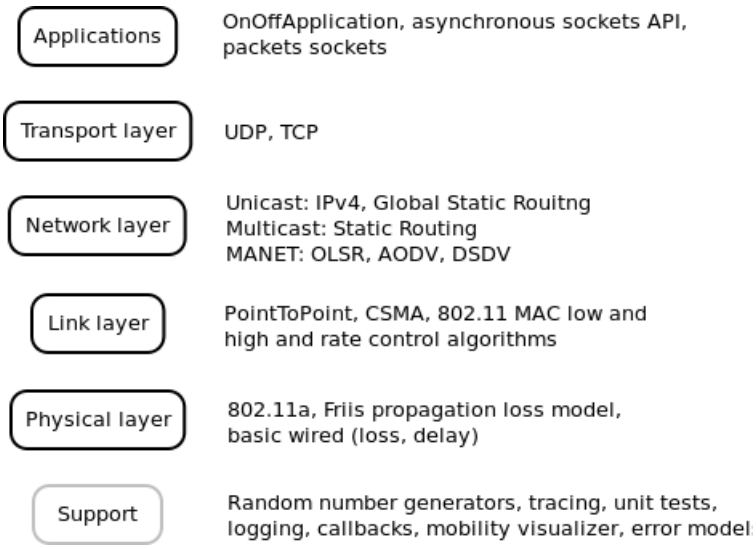


Figure 4.1: Some of the different models implemented in the ns-3.10 release of the simulator.

The architecture and structure of the ns-3 simulator is further explained in Chapter 5.

4.3 Related Simulation Studies

Both simulators described above have been used to evaluate a range of different routing protocols tailored for MANETs [Ver11].

However, support for the BATMAN routing protocol has not yet been implemented

in any network simulators such as ns-2 and ns-3 [Mesaya]. Thus no simulations with this protocol have been published at the time of writing.

Since no simulator have yet to add support to this protocol, the decision fell on using ns-3 due to it being a newer and more modern simulator developed by people who knew the advantages and issues of ns-2.

Part II

BATMAN ns-3 modules

Chapter 5

Implementation

Two new routing models were implemented in ns-3 routing module including the original BATMAN and a simplified version of the modified BATMAN protocol.

This chapter covers how the ns-3 network simulator is organized and how implementation of these two protocols fit into this structure. The chapter also describes in more detail how they were implemented including explanations of the most important functions, classes and subclasses as well as their main attributes and components.

The Destination Sequenced Distance Vector (DSDV) protocol is the newest ad hoc routing protocol added to the latest stable release ns-3.10 at the time of writing [NCÇ+11]. This protocol and its model in ns-3 was used as a source of inspiration when implementing the BATMAN routing protocol. Otherwise, the implementation of BATMAN follows the BATMAN standard draft proposed in [NALW11].

5.1 ns-3 Architecture

Figure 5.1 shows a schematic view of how the source code in ns-3 is organized.
The figure shows that the simulator is divided into separate modules where the modules only have dependencies to other modules placed beneath them. A ns-3 module is built as a separate software library where ns-3 programs can link the modules they need to conduct their simulation. A module may consist of one or more models which are abstract representations of real-world object, protocols, devices etc. [nsnaya].

The core of the simulator consists of the three modules `core`, `common` and `simulator`. Together they create a generic simulation foundation which is common across all protocol, hardware, and environmental models making it usable for any kind of network, not just only Internet-based (IP-based) networks.

The two modules in the top layers of Figure 5.1, `helper` and `test`, are supplements to the C++ core API. ns-3 programs, also referred to as simulations scenarios or

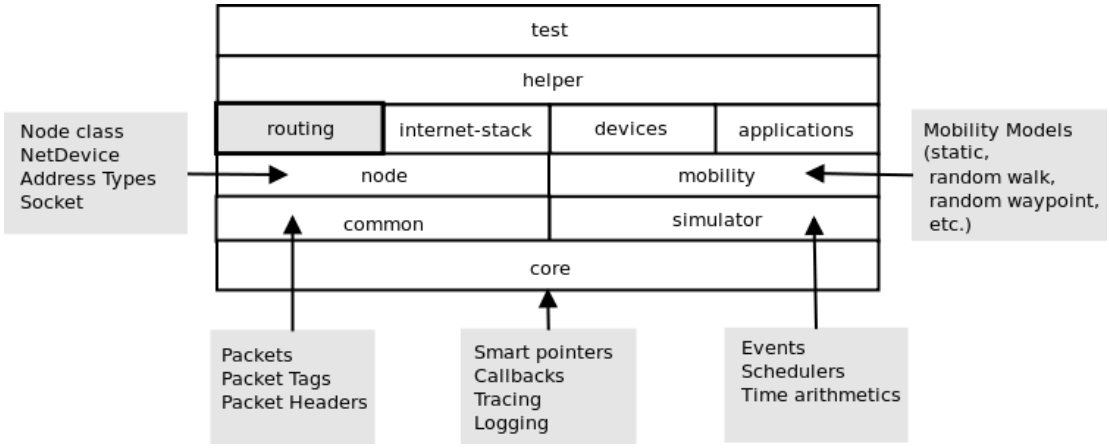


Figure 5.1: Software organization of the ns-3 simulator, based on the figure from [nsnayb].

scripts, can access the core API directly or use the high-level wrappers and encapsulations found in the `helper` module [nsnayb].

The other layers in Figure 5.1 add the networking-specific components of ns-3. For instance, the `internet-stack` module includes implementations of protocols such as User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) while the `routing` module provides different models of IPv4 routing protocols.

The organization of ns-3 provides a modular source code where different models can be added to a module without having to make changes to other modules or the entire ns-3 architecture. The BATMAN routing protocol was implemented in the `routing` module which is highlighted in Figure 5.1. A BATMAN `helper` class was also implemented to assist when using the BATMAN protocol in simulation scenarios.

The next section briefly explains how the protocol was implemented including its main classes and attributes.

5.2 Class Interaction

The relations between the BATMAN classes implemented in the `routing` module, are presented in Figure 5.2.
As the figure illustrates, `ns3::batman::RoutingProtocol` is a subclass of the already implemented abstract base class `ns3::Ipv4RoutingProtocol`. This class performs the main routing tasks and contains functions that connect the routing model to the rest of the ns-3 core.

The `ns3::batman::OriginatorList` class contains of a collection of Originator List Entries declared in the `ns3::batman::OriginatorListEntry` class. These entries

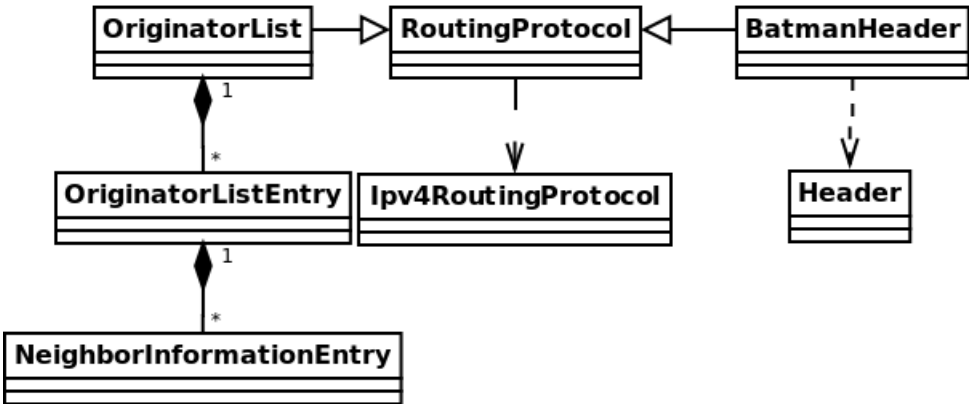


Figure 5.2: Class Diagram for the BATMAN model in ns-3 omitting attributes and methods due to size issues.

are equivalent to the originator list entries as explained in Chapter 2. Every originator list entry also has a neighbor information list which includes a set of Neighbor Information Entries declared in the `ns3::batman::NeighborInformationEntry` class.

The `ns3::batman::BatmanHeader` class implements the packet format and it is extended from the existing `ns3::Header` base class. The details of all these classes are explained in the sections below.

5.3 Originator Message OGM

The `ns3::batman::BatmanHeader` class defines the format of the BATMAN packet which consists of one OGM and zero or more optional HNA extension messages. The total size of an OGM header is fixed at 20 bytes while the total size of an entire BATMAN packet depends on how many HNA extension messages that are appended. However, as discussed in Section 2.2.1, the HNA functionality is not implemented in ns-3.

The BATMAN packet is encapsulated inside a UDP datagram as depicted in Figure 5.3.

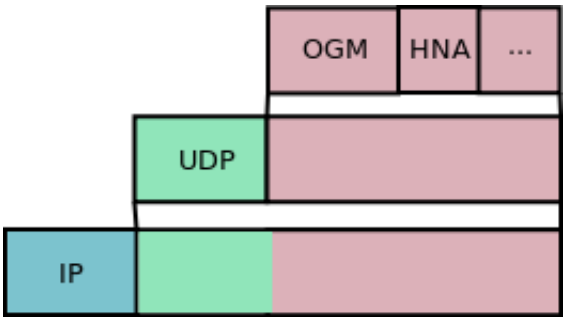


Figure 5.3: A BATMAN packet encapsulated in a UDP datagram.

The OGM without the optional trailing HNA messages, was implemented after the format from BATMAN IV explained in Section 2.3 and is illustrated in Figure 5.4.

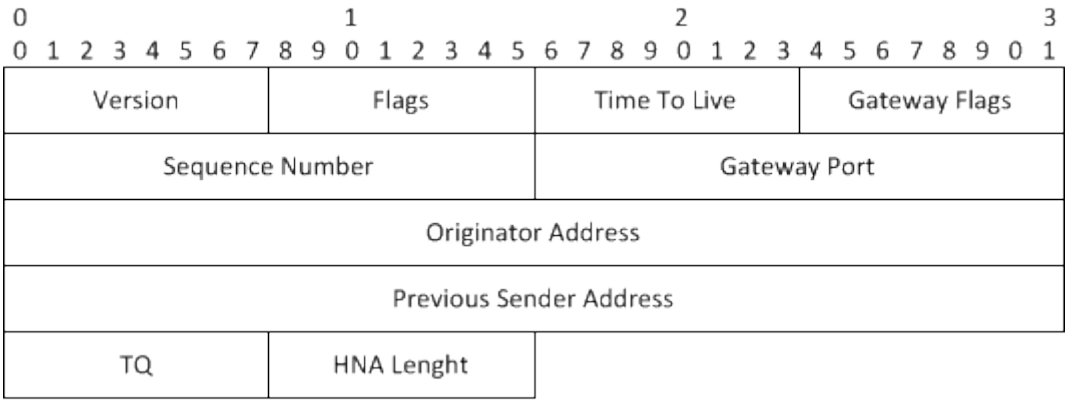


Figure 5.4: Format of the OGM excluding the optional HNA messages.

The fields that are not used, Gateway and HNA, are not omitted, just set to zero when running the protocol.

The `ns3::batman::BatmanHeader` class includes two methods inherited from the abstract base class `ns3::Header` called `Serialize()` and `Deserialize()`. These methods convert the BATMAN header into a byte buffer in its network representation.

5.4 Originator List

A node's Originator List is equivalent to an ordinary routing table and it is maintained in the `ns3::batman::OriginatorList` class as a C++ map. This class includes methods to add, delete, update, purge and look up entries from the Originator List.

A node stores an entry in its Originator List for every originator from which or via which it has received an OGM just as described in Section 2.2.2 and 2.2.4. These Originator Entries are defined in the `ns3::batman::OriginatorListEntry` class and holds the following information:

- **Originator Address**
- **Current Sequence Number**
- **Last Aware Time**
- **Next Hop**
- **Last Time To Live**
- **Neighbor Information List**

Every Originator in a nodes Originator List may be reached via several link-local neighbors. These neighboring nodes are added as entries in the Neighbor Information List and the entries are declared in `NeighborInformationEntry` class and contains the following attributes:

- **Neighbor Address**
- **Last Valid Time**
- **Last Time To Live**
- **Sliding Window for rebroadcasts of own OGMs (EQ)**
- **Sliding Window for total received OGMs from this neighbor (TQ)**

This list is used by the protocol to choose the best next hop towards the originator to which it belongs as described in Section 2.2.2. When and how this selection is don, is explained in the next section.

5.5 BATMAN Routing Protocol

The main routing logic is performed in the `ns3::batman::RoutingProtocol` class making it the most important component in the BATMAN implementation. The class inherits functions from its parent class such as `RouteInput` and `RouteOutput` which tie together the network layers in the simulator. It also combines functions from the other classes to build and maintain the Originator List.

The main functions in this class are `BroadcastOriginatorMessage`, `RecvOriginatorMessage`, and `ReBroadcastOriginatorMessage`.

`BroadcastOriginatorMessage` function is in charge of periodically broadcasting OGMs on behalf of the node as explained in Section 2.2. Is keeps a `OriginatorIntervalTimer` which schedules the events where an OGM is to be broadcasted.

`RecvOriginatorMessage` handles the processing of received OGMs. The most important checks an OGM go through is `BidirectionalCheck` and `DuplicateCheck`. These checks verify that the OGM is from a bidirectional neighbor and not a duplicate packet. During the checks the received sequence numbers are registered, sliding windows are updated and calculations are performed updating the EQ and TQ values as described in Section 2.3.

If the two check are passed, the function `UpdateNeighborRanking` does as the name implies, update the neighbor ranking choosing the best neighbor as the next hop towards the originator of the OGM.

Finally, the OGM is rebroadcasted if it has passed all the checks and has a TTL value is more than zero. `ReBroadcastOriginatorMessage` is in charge of rebroadcasting OGMs and resembles the `BroadcastOriginatorMessage` function just that it is called on request, not periodically.

5.6 Routing Attributes and Default Values

Table 5.1 shows the different attributes and their default values used in the implementation.

Attribute	Defaults	Description
DEFAULT_VERSION	4	BATMAN version
MAX_SEQUENCE_NUMBER	65535	Maximum sequence number allowed
DEFAULT_TTL	50	Default Time To Live value
OGM_BROADCAST_INTERVAL	1 s	Periodic interval between broadcasting of OGMs
TQ_LOCAL_WINDOW_SIZE	64	Size in bytes of the sliding window used when calculating TQ values.
PURGE_TIMEOUT	200 s	Lifetime of an Originator List Entry which has not been updated recently.
HOP_PENALTY	5	Penalty added to an OGMs TQ value for every hop made in the network.
MAX_TQ_VALUE	255	Maximum TQ value possible.

Table 5.1: BATMAN attributes and their default values.

5.7 BATMAN Security Elements

After implementing the original BATMAN protocol, this model was used as a base when implementing the modified version which will be referred to as Secure BATMAN. Security elements such as encryption tools and key functions were based and inspired by on the ongoing work done by Espen Grannes Graarud on his master thesis at NTNU which involves implementing a real life version of the secure BATMAN protocol [1].

The main focus when implementing this version of BATMAN was to capture the security aspects added which affect the total routing performance the most. The added workflow of the original BATMAN can be roughly summarized in three main steps:

1. PC signing and issuing,
2. verification of other nodes' PCs, and
3. continuous authentication of received OGMs

Otherwise, the Secure BATMAN protocol behaves approximately the same as the original version.

[1] Can be found here ...

Obviously it is the last step which is done continuously during the routing and thus put the most average and continuous work on the protocol.

Also, assuming that a node is issued a PC which is valid during an entire e.g. rescue operation this is done once at the very beginning. Thus an added delay could vouch for this added time. In addition, the exchanging and verification of nodes' PC1s is also done once and afterwards the nods only need to get new and key stream material to generate key streams. These are the task done the most and thus affect the overall routing performance the most. Thus this were prioritized when implementing this version.

The following section describes the most important functionality added to the original BATMAN.

5.7.1 Packet Format

As explained in Section 3.7, the OGM was extended by adding two new fields in order to include the authentication mechanisms shown in Figure 3.4.

As the previously, the the packet format is defined in `ns3::securebatman::SecureBatmanHeader` which is extended from `ns3::Header`.

5.7.2 Initial Values and Key Stream Generation

The most important functions added in this model is `GenerateInitialValues`, `GenerateKeystream`, `GenerateNeighborKeystream` and `CheckOneTimePassword`. The first step done by a node running the secure BATMAN protocol, is to generate it's ephemeral key, nonce and IV values which is afterwards transmitted to its link-local neighbors.

The ephemeral key if created using xxx from the openssl library while the nonce and IV values are only random numbers. All the function involving security generation resides in the new class `ns3:securebatman:am` which is an abbreviation for Authentication Module.

When the AM packet has been sent to link-local neighbors, the node uses `GenerateKeystream` function to generate it's own key stream. Every OGM broadcasted from then on contains an 16 bit extract from this stream along with a sequence number as explained in Section xx.

In the `RecvOriginatorMessage` function additional checks were added to verify the the one time password contained in the new Authentication Fields of an OGM. `CheckOneTimePassword` simply checks to see if the neighbor has been added to it's list with a corresponding key stream. If it exists, then the One Time password

retrived from the OGM is check and the OGM is further process as normal if it is verified.

If an AM packet has been recevied, then the receiver will use the function `GenerateNeighborKeystream` to generate and store the key stream of the neighbor.

5.7.3 OpenSSL

The basic purpose of the of the SSL protocol and its succssesor, TLS: provide the most common security services to arbitrary (TCP-based) network connections in such a way that the need for cryptographic expertise is minimized. The OpenSSL is a cryptographic library; it provides implementations of the industry's best-regarded algorithms, including encryption algorithms such as 3DES, AES and RSA as well as message digest algorithms and message authentication codes. strives toward that ideal [VMC02]

Part III

Simulation and Results

Chapter 6

Simulation Setup

The previous chapter explained how both versions of the BATMAN protocol are implemented as routing modules in ns-3. This chapter explains the different simulations scenarios where these routing modules are used as well as the different metrics that were measured during the simulations.

6.1 Performance Metrics

The goal of the simulations is to measure both the original and the modified BATMAN protocol's ability to react to network topology changes while still continuing to successfully deliver packets efficiently to a destination.

This ability is defined by a selection of performance metrics which is measured during the simulations and then analyzed and compared afterwards. The following metrics are valuable when evaluating the performance of a routing protocol [CBD02, BMJ+98]:

- **Packet Delivery Ratio (PDR):** The amount of packets received divided by the amount of packets actually sent by the application layer.
- **Routing Overhead:** The total amount of routing packets (OGMs) transmitted during the entire simulation.
- **End-to-end Delay:** The time taken by a packet when transmitted from a source to a destination measured at the MAC layer.

The PDR metric gives an indication of the protocol's loss rate which affects the maximum throughput that the network can support. Routing overhead shows the scalability of the protocol while the transmission delay indicates how efficient the protocol is when choosing the best path to a destination. All these metrics will me monitored and measured for the evaluation of the routing protocols afterwards.

6.2 Mobility Models

Mobility models describe the movement of a node during a network simulation. This includes the direction of movement, velocity and acceleration of the node over time.

Several mobility models exist for wireless ad hoc networks in ns-3 and the most relevant models for the simulations performed in this report, are [CBD02]:

- **Random Walk Mobility Model:** Describes mobility patterns were a node moves from its current position to a new random destination moving with random speed.
- **Random Waypoint Mobility Model:** Mobility model which also includes pause times between the changes in destination and speed.
- **Random Direction Mobility Model:** Describes node movement patterns where the nodes are forced to move to the edge of the simulation area before changing direction and speed.

The models mentioned above are all entity mobility models which define the movement of the nodes independently of each other. The two first models are the most common mobility models used by researchers [CBD02].

The mobility model utilized in the simulations performed this report, is the Random Waypoint Mobility Model. Even though it is one of the most commonly used models in ad hoc research, it is proved that it suffers from certain issues that might affect the performance results from simulations [BRS03, CBD02]. However, this issue can be avoided by following some simple recommendations e.g. discarding the initial simulation time produced by the model [CBD02]. That is one of the reasons why the nodes in the simulations performed here wait a certain "settling time" before sending traffic in the network. This is further explained in Section 6.3.

6.3 Methodology and Simulation Setup

Running a simulation in ns-3 works by writing a program which describes the simulation scenario with all its models used, node movement and environments. A simulation scenario can be run with different parameters creating even more

The different parameters used to describe the simulation scenario as well as the methodology of how the simulations were executed, are inspired by previous studies involving simulations with other ad hoc routing protocols [NCÇ+11, BMJ+98, DPR00].

6.3.1 Physical Space and Node Movement

The simulations are performed using a varying amount of nodes, 10, 20 and 30, which are moving in a rectangular flat space (1500x300m^2).

The nodes movement is as mentioned described by the `RandomWaypointMobility` model already implemented in ns-3. The nodes' velocity is set to vary between 0 and 20 m/s during a simulation run and every run is done with 10 different pause times varying from 0 to 900 seconds.

The total simulation time for every simulation scenario is set to 900 seconds. This means that the simulations scenario running with pause time 0 seconds entails that nodes are moving continuously with no stops. When using a pause time of 900 seconds, the nodes stay at a fixed position the entire simulation run.

6.3.2 Traffic Generation and Flows

All nodes transmit and receive constant data traffic to and from other nodes in the network. Based on the previous studies, a packet size of 64 bytes was chosen due to very low PDRs values when using a larger packet size (>500 bytes) [NCÇ+11, BMJ+98]. The nodes are configured to generate Constant Bit Rate (CBR) traffic using the `OnOff` application in ns-3 with a data rate of 4 packets/s (256 Bps). The simulator is configured to create as many traffic flows as the amount of nodes in the network and it is ensured that every node both receives and transmits data.

Nodes generate CBR traffic using 802.11b MAC over the Friis propagation loss model [Fri46] to limit the nodes transmission range [NCÇ+11, BMJ+98].

All the simulation parameters that are constant for every simulation run, is summarized in Table 6.1.

Parameter	**Value**
Node Velocity	0 - 20 m/s
Packet Size	64 bytes
Data Rate	256 Bps
Settling Time	30 s
Simulation Time	930 s
BATMAN OGM Broadcasting Interval	1 s

Table 6.1: Simulation parameters that stay constant for every simulation run.

6.3.3 Simulation Statistics and Data Collection

Simulation metrics collected and analyzed are averaged over 10 repeated simulation runs to get an central tendency of the data set. This sums up to 300 simulation runs with different combinations of nodes and pause times as illustrated in Table 6.2.
The simulation scenario contains several trace sinks connected to trace sources in the ns-3 core and data calculators which gets data which is collected by a data collector

Repetitions	Node Count	Pause Times (s)
10 x	10	0 100 ... 800 900
	20	0 100 ... 800 900
	50	0 100 ... 800 900

Table 6.2: Illustrations of the different parameter combinations used and amount of repetitions of the simulation runs.

during a simulation run. After the simulations run is completed, the datacollector is in charge of storing the data in a SQLite database for post-processing and analysis.

6.4 Running the Simulations

The simulations are controlled by a simulation script which launches the different simulation scenario with the different parameter combinations. It is also in charge of querying the SQLite database and parsing results for creating graphs with GNUplot.

The order of the steps performed by the simulation script is described in the list below and illustrated in Figure 6.1.

1. Start simulation script
2. Script runs the simulation scenario written in C++ specifying the different parameters to be used (pause time, amount of node etc.)
3. Simulation scenario measures and store data from the simulation in the SQLite database
4. Simulation script queries the database ans store data point in a file
5. Simulation script invokes GNUplot to create the graphs from the datapoints received and parse from the database

The Figure 6.1 illustrates the workflow of the simulation script.
The script along with the simulation scenario written in C++ can be found in Appendix E and D respectively.

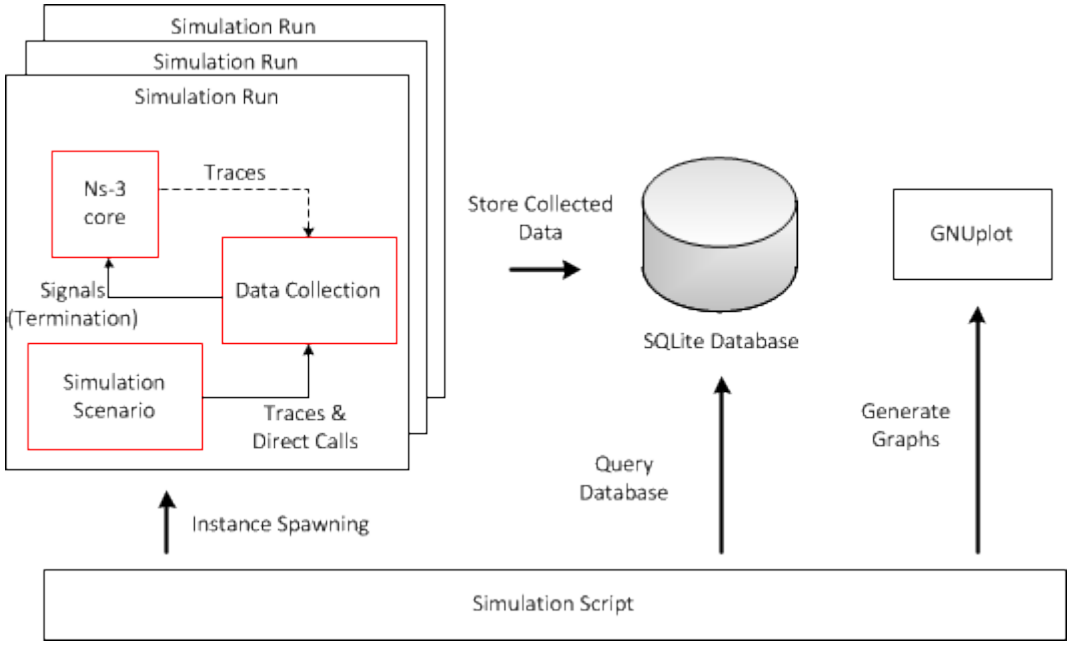


Figure 6.1: Workflow and involved entities during the lifetime of the simulation script.

6.5 Simple Performance Comparison of DSDV and BATMAN

BATMAN's main principle is that routing information is flooded through the network and thus shared amongst every single participating node. Every node broadcasts routing information which is rebroadcasted through the network until all the other nodes have received the information. This creates a lot of traffic and increases the possibility of packet collisions and interference. Also network scalability becomes an issue.

The developers justify this design choice with the fact that wireless networks are by nature very lossy and a high packet loss is expected [NALW11]. Thus the simplest and most efficient way of maintaining the network, is to simply flood it with routing information and expect/assume that a large majority of the packets will be lost.

DSDV is also a proactive routing protocol, but it produces however way less routing overhead compared to BATMAN. Thus, it would be interesting to investigate how the performance of this protocol is compared to BATMAN when exposed to a more lossy wireless environment.

This comparison was done by performing the same simulations as explained above, but decreasing the nodes' transmission power to reduce their transmission range. This creates weaker links between nodes making a more unstable network.Thus the protocols were tested in what is referred to as one "strong network" and one "weak

network".

DSDV was run with a Settling Time set to 6 seconds and something else set to 15 seconds.

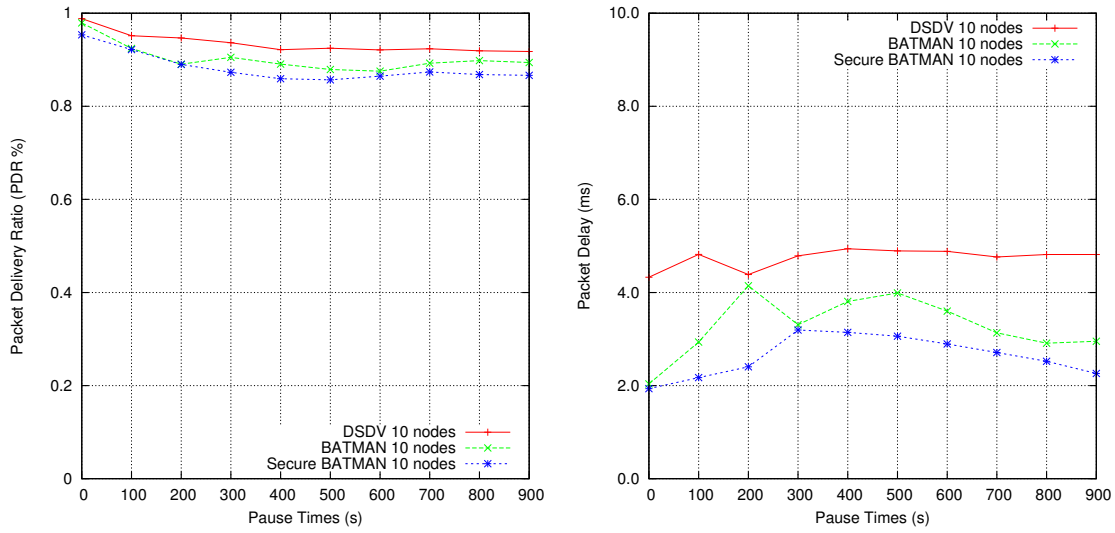
The results from the simulations perform together with a analyzis is presented in the next chapter.

Chapter 7

Simulation Results

This chapter presents the results from the various simulations scenarios described in the previous chapter.

Figure 7.1 shows Packet Delivery Ratio (PDR) and packet delay results from the simulations running the Extended BATMAN, BATMAN and Destination Sequenced Distance Vector (DSDV) with 10 nodes and 10 traffic flows.



(a) PDR with varying pause times.

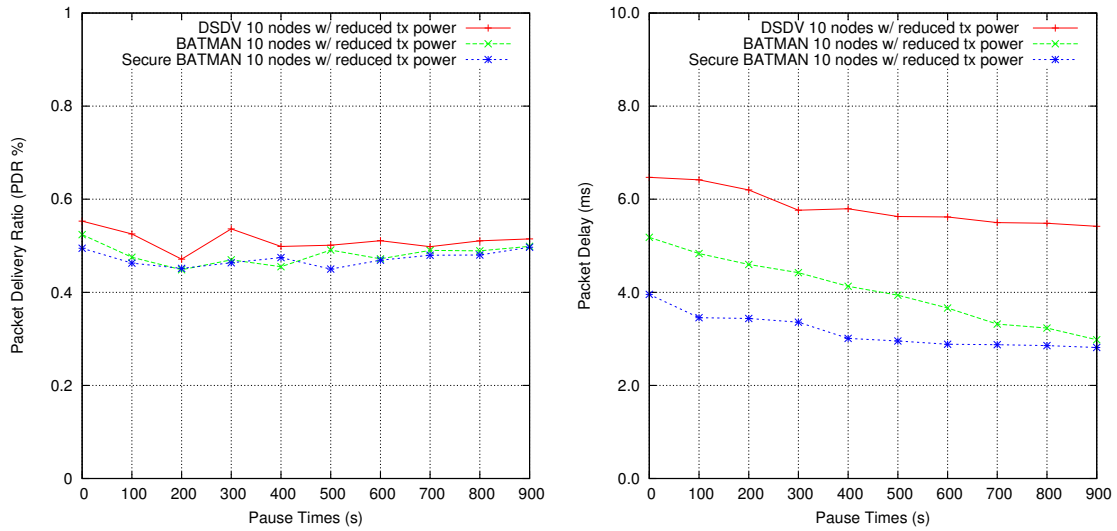
(b) Packet delay with varying pause times.

Figure 7.1: Results from BATMAN, Secure BATMAN and DSDV running with 10 nodes and 10 source and sink pairs.

As seen from the graphs in Figure 7.1a, the PDR values of all three routing protocols stay well above 80%. Interestingly, the Extended BATMAN protocol's PDR also stay at approximately the same level as the two other protocols. At pause time zero, which is equivalent to continuous node movement, all three protocols show their best behavior with the highest PDR values.

When looking at the measured/average packet delay in Figure 7.1b it is surprisingly the Extended BATMAN protocol which has the lowest values.

As mentioned in 6.5, the same simulations were run with decreased transmission power of the nodes in order to create a weaker network. The Figure 7.2 shows the PDR values and packet delays derived from the simulations results.



(a) PDR with varying pause times.

(b) Packet delay with varying pause times.

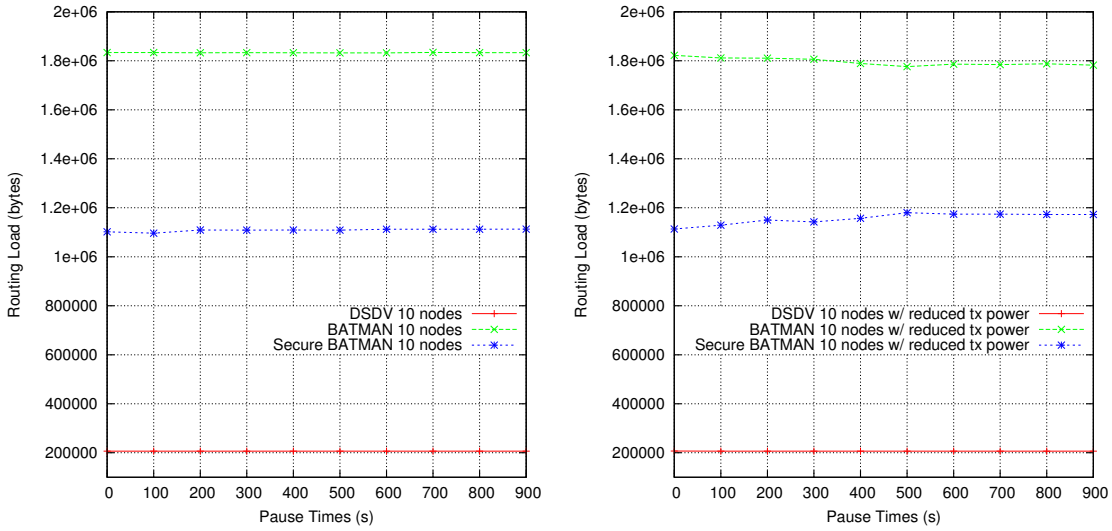
Figure 7.2: Results from BATMAN, Secure BATMAN and DSDV running with 10 nodes and 10 source and sink pairs where the transmission power has been reduced.

From reducing the transmission power the PDR values drop significantly as shown in Figure 7.2a. This is due to the fact that packets no longer reach as far as in the previous scenario and the routing overhead create more collisions and interference since the signals are weaker. Still all three protocols preform almost equally well when delivering packets from source to destination.

The packet delays presented in Figure 7.2b of all the three protocols are slightly increased in this scenario. This is natural as the packets probably have to use longer paths (more hops) to arrive at the destination since the signals are weaker.

Figure 7.3 shows the routing load in bytes produced by the three protocols during both of two first scenarios, with and without reduced transmission power.
As expected, both BATMAN and Secure BATMAN create way more load on the network compared to DSDV. However all three protocols create an approximately constant load on the networks in both scenarios.

The next simulation scenarios conducted increased the number of nodes as well as the amount of traffic flows. Figure 7.4 presents the PDR results from both BATMAN and DSDV with 20 nodes both without and with reduced transmission power

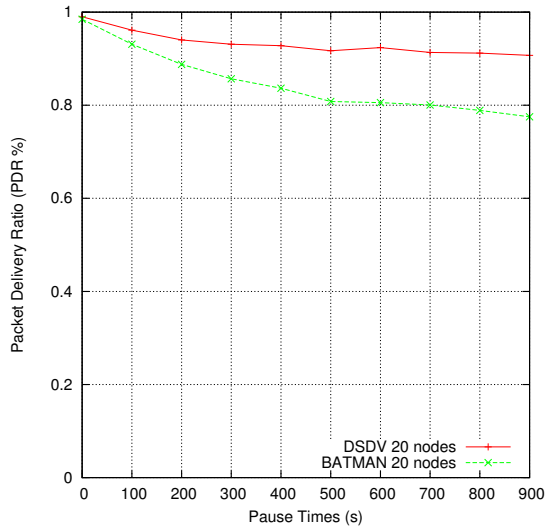


(a) Routing overhead with varying pause times. (b) Routing overhead with varying pause times and reduced transmission power.

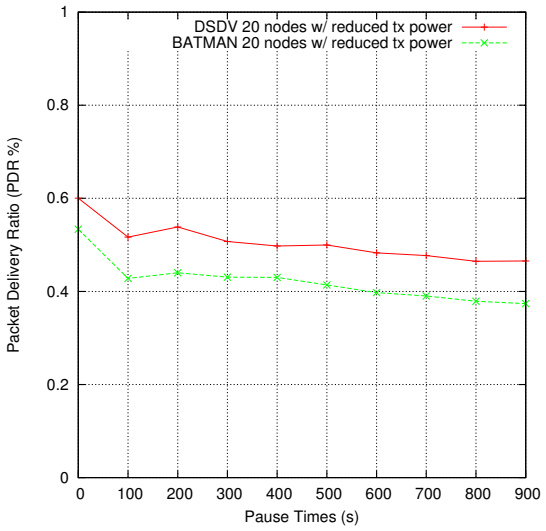
Figure 7.3: Results from BATMAN, Secure BATMAN and DSDV running with 10 nodes and 10 source and sink pairs where the transmission power has been reduced.

respectively. Due to the reasons justified in Section XX, Secure BATMAN was not included in these simulations.
The situation is still the same here as in with DSDV being slightly better at delivering packets than BATMAN in both scenarios.

Finally, simulations were also run with 30 nodes both in a ”strong network” and ”weak network”. Results are summarized in Figure 7.5.
The graphs are very similar to the one presented in Figure 7.4. This implies that neither of the protocols are very affected by the increasing amount of nodes and traffic flows.

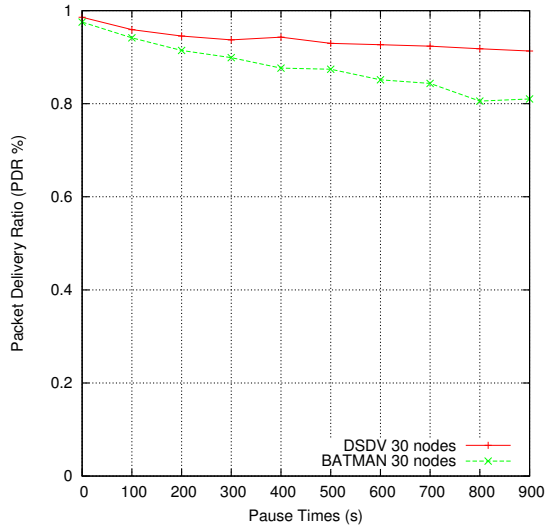


(a) PDR with varying pause times.

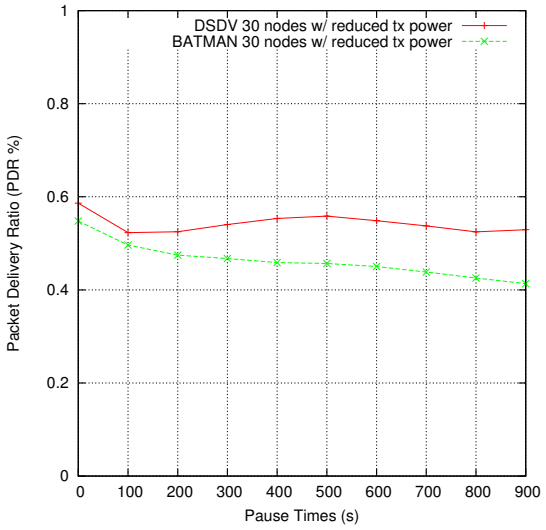


(b) PDR with varying pause time and reduced transmission power.

Figure 7.4: Results from BATMAN, Secure BATMAN and DSDV running with 20 nodes and 20 source and sink pairs.



(a) PDR with varying pause times.



(b) PDR with varying pause time and reduced transmission power.

Figure 7.5: Results from BATMAN, Secure BATMAN and DSDV running with 30 nodes and 30 source and sink pairs.

Chapter 8

Discussion

This chapter discusses some of the results presented in the previous chapter pointing out important aspects they highlight. The chapter also discusses protocol validation and the experience of working with the ns-3 simulator. A short evaluation of the security of the Secure BATMAN protocol is also included.

8.1 Performance Results

According to the results presented in the previous chapter, the Secure BATMAN routing protocol does not perform significantly worse than it's original counterpart or DSDV.

The protocol's PDR values are in both scenarios, "weak network" and "strong network", at the same level as the original BATMAN and DSDV. It indicates that it's ability to route and deliver application data is not significantly affected by the added security elements. The extra checks and computations performed when receiving one-time passwords and generating key streams etc. does not affect the total delivery rate/ability of the protocol.

The Secure BATMAN protocol actually gives the lowest packet delay during both simulation scenarios. However, since the packet delay is measured at MAC level this entails that also routing protocols are registered. Thus the average value measured for the Secure BATMAN protocol is likely to be reduced due to the transmissions of the extra AM packets as described in Section 5.7.

All three routing protocols are proactive meaning that routing information transmitted in the network will be significant as mirrored in the results. As expected, both BATMAN and the Secure BATMAN impose way more load on the network compared to DSDV. However, this added routing load does not affect neither the protocols' PDR or average packet delay.

The Secure BATMAN has less routing overhead than the original BATMAN. This is probably due to the periodic exchange of AM packets between nodes which reduces

the total amount of BATMAN packets which are flooded through the network. Thus the average routing overhead produced by the Secure BATMAN protocol is reduced.

However, despite their vastly different amounts of overhead, all three protocols have routing loads that stay nearly constant in both scenarios also with varying pause times. This is expected due to the fact they are proactive routing protocols which means that the amount of routing information transmitted is not affected by changes in the topology or network environment and behavior such as transmission power and pause times.

The original BATMAN protocol was further tested and compared against DSDV in scenarios containing more nodes and traffic sources. Simulations were done in both ”strong network” and ”weak network”. The results from these simulations show that DSDV perform slightly better than BATMAN.

It was expected that the BATMAN protocol probably would perform better or at least the same as DSDV in a network which was weaker and had lossy links. According to the behavior mirrored in the PDR measured during the simulations, this is not the case. BATMAN is always slightly below DSDV.

Not all aspects of the Secure BATMAN routing protocol were implemented in ns-3 as explained in Section 5.7. Also, the protocol was not tested in environments with more nodes and more traffic. However, since the protocol in the first simulations did perform similar to its original counterpart, we can assume that it would behave identically in these situations as well.

8.2 Security Design Choices of the Secure BATMAN

As Section 3.2 describes, the main goal of the security measures added to the BATMAN protocol, was to include a form of access control mechanism in As (MANETs) which were to be used in emergency situations and similar. This were to be included without impacting the overall performance and functionality of the protocol.

The proposed design does present a solution which accomplishes this. However, the design does have some weaknesses and issues which should be assessed, some of which include:

- Security weaknesses
- Network Merging
- Procedure of Proxy Certificate (PC) issuing
- Service Proxy (SP) presence

- Multiple SPs

These issues were discussed in the specialization project [Neeay] and solutions to how they could be solved were proposed. //MORE

8.3 Protocol Validation

As mentioned in Section 4.3, the BATMAN protocol has not yet been implemented in any network simulator at the time of writing. Thus, there were no possibility of validating the behavior of the protocol implemented in ns-3 except from comparing it to the real life protocol.

So, the behavior of the implemented protocol was verified by manually studying and comparing detailed verbose debug output from both the real life version and the protocol implemented in ns-3. Seen from these tests the protocols behave identically calculating the correct values used during routing and making the same routing decisions. As small excerpt from some of these debug outputs used to validate the protocol behavior is added to Appendix XX.

8.4 Experience Working with ns-3

ns-3 presents itself as being a powerful simulator with great opportunities and a modularity which makes it relatively easy to extend.

However, the simulator still has a steep learning curve. Partly due to it's higher degree of realism the complexity of the simulator is also increased compared to ns-2. Low-level details are introduced at an early stage of the implementation process increasing the time and effort needed to get to know the simulator in order to introduce new elements. However, when this initial learning curve is conquered, the simulator shows it's wide range of possibilities. In addition it's framework for data and statistics collection as well as experiment control is good.

Due to its complexity and lack of supported models, the opportunity to quickly test and study research ideas is not as trivial. In many ways ns-2 would be preferred in these situations having implemented support for more models as well as being "easier to use".

ns-3 is however actively supported and developed thus the simulator should evolve and include more models and increased documentation and tutorials in the future.

Running simulations is a time consuming and resource demanding task. It was expected to be able to run more simulations than what was actually done, but due to time and resource constraints this was not possible. This took longer than expected. And doing several changes in between long simulation runs was not an option.

Chapter 9

Conclusion

The goal of the work behind this report was to extend the network simulator ns-3 in order for it to support the BATMAN ad hoc routing protocol and a modified version of the same protocol which included security elements for a authentication mechanism. Both implementations were then to be used to conduct simulations evaluating and comparing the protocols' performance in various scenarios. Both protocols were also compared to the already existing DSDV protocol in ns-3 to evaluate their general performance as well.

Before the implementation was started, a comprehensive study was done to investigate the different network simulators available. An effort was also put in to understand the background of the most prominent network simulators including their advantages and limitations.

This report presents the implementation of the BATMAN protocol in ns-3 including details about the different components, important attributes and class interactions. A description of how difference openssl tools are included for the security elements are also present.

Due to time constraints and the time consuming side effect of running the simulations, not all aspects of the Extended BATMAN protocol was implemented. However, what was considered the most important elements with respect to the affect it would have on the total routing performance, was added to the routing model in ns-3.

The results derived from the simulations indicate that the routing performance of the Extended BATMAN protocol is not substantially affected by its security extensions.

Other interesting results from the simulations performed, is the BATMAN protocol does not perform as well or better in the environments in which it should be superior.

9.1 Further Work

Future work would include spending more time validating the BATMAN protocol implemented in ns-3.

Also, the remaining security elements should be added to the Extended BATMAN protocol in ns-3 and additional simulations should be conducted.

References

[BBE+99] Sandeep Bajaj, Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, Padma Haldar, Mark Handley, Ahmed Helmy, John Heidemann, Polly Huang, et al. Improving Simulation for Network Research. Technical report, Technical Report 99-702b, University of Southern California, 1999.

[BEF+02] Lee Breslau, Deborah Estrin, Kevin Fall, Sally Floyd, John Heidemann, Ahmed Helmy, Polly Huang, Steven McCanne, Kannan Varadhan, Ya Xu, and Haibo Yu. Advances in Network Simulation. *Computer*, 2002.

[BG10] Anne Gabrielle Bowitz and Espen Grannes Graarud. Developing a Secure Ad Hoc Network Implementation, 2010.

[BMJ+98] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. ACM, 1998.

[BRS03] Christian Bettstetter, Giovanni Resta, and Paolo Santi. The node distribution of the random waypoint mobility model for wireless ad hoc networks. *IEEE Transactions on Mobile Computing*, 2003.

[CBD02] Tracy Camp, Jeff Boleng, and Vanessa Davies. A Survey of Mobility Models for Ad Hoc Network Research. *Wireless communications and mobile computing*, 2002.

[CJ10] Thomas Heide Clausen and Philippe Jacquet. Optimized Link State Routing Protocol (OLSR). *Network Working Group*, Last accessed December 19, 2010. http://tools.ietf.org/html/rfc3626.

[Dir07] National Police Directorate. *Police Emergency Prepareness System, Part I Norway - Emergency Manual*. someone, 2007.

[DPR00] Samir R. Das, Charles E. Perkins, and Elizabeth M. Royer. Performance comparison of two on-demand routing protocols for ad hoc networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*. IEEE, 2000.

[FKGay] Sheila Frankel, Scott Kelly, and Rob Glenn. The AES-CBC Cipher Algorithm and Its Use with IPsec. *Network Working Group*, Last accessed June 24, 2011. http://www.faqs.org/rfcs/rfc3602.html.

[Fri46] Harald T. Friis. A note on a simple transmission formula. *Proceedings of the IRE*, 1946.

[GKS95] Nada Golmie, Alfred Koening, and David Su. *The NIST ATM network simulator: Operation and programming*. National Institute of Standards and Technology, 1995.

[HPFS02] Russell Housley, Tim Polk, Warwick Ford, and David Solo. Internet x. 509 public key infrastructure certificate and certification revocation list (crl) profile. *Network Working Group*, 2002. http://tools.ietf.org/html/rfc3280.

[HRFR06] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. ns-3 Project Goals. In *Proceeding from the 2006 Workshop on ns-2: The IP Network Simulator*, 2006.

[HRFRay] Thomas R. Henderson, Sumit Roy, Sally Floyd, and George F. Riley. *Developing the Next-Generation Open-Source Network Simulator (ns-3). http://www.nsnam.org/docs/meetings/snowbird06/ns-3-cri-workshop-2006.pdf*, Last accessed June 24, 2011. http://www.nsnam.org/docs/meetings/snowbird06/ns-3-cri-workshop-2006.pdf.

[Kes11] Srinivasan Keshav. *REAL 5.0 Overview. http://www.cs.cornell.edu/skeshav/real/overview.html*, Last accessed Februar 28, 2011. http://www.cs.cornell.edu/skeshav/real/overview.html.

[Mes11] Open Mesh. *BATMAN Documentation. http://gitorious.org/batman-adv-doc*, Last accessed May 4, 2011. http://gitorious.org/batman-adv-doc.

[Mesaya] Open Mesh. *FAQ. open-mesh.org*, Last accessed June 24, 2011. http://www.open-mesh.org/wiki/open-mesh/FAQ.

[Mesayb] Open Mesh. *The OLSR Story. open-mesh.org*, Last accessed June 24, 2011. http://www.open-mesh.org/wiki/the-olsr-story.

[NALW11] Alex Neumann, Corinna Aichele, Marek Lindner, and Simon Wunderlich. Better Approach To Ad-Hoc Networking (B.A.T.M.A.N) draft-wunderlich-open-mesh-manet-routing-00. *Network Working Group*, Last accessed Februar 28, 2011. http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00.

[NCÇ+11] Hemanth Narra, Yufei Cheng, Egemen K. Çetinkaya, Justin P. Rohrer, and James P.G. Sterbenz. Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3. *something*, 2011.

[Neeay] Citation Needed. Citation Needed. *Something*, Last accessed June 24, 2011.

[nsnaya] nsnam. *ns-3 Model Library. http://www.nsnam.org/*, Last accessed June 24, 2011. http://www.nsnam.org/docs/release/3.11/models/ns-3-model-library.pdf.

[nsnayb] nsnam. *ns-3 Tutorials and Manual. http://www.nsnam.org/tutorials.html*, Last accessed June 24, 2011. http://www.nsnam.org/tutorials.html.

[nsnayc] nsnam. *Official ns-2 Website. http://www.isi.edu/nsnam/ns/*, Last accessed June 24, 2011. http://www.isi.edu/nsnam/ns/.

[PB94] Charles E. Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (dsdv) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 1994.

[PBRD03] Charles Perkins, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (aodv) routing. *Network Working Group*, 2003.

[RSA78] Ronald Linn Rivest, Adi Shamir, and Leonard Max Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[SQLay] SQLite. *About SQLite. http://www.sqlite.org/about.html*, Last accessed June 24, 2011. http://www.sqlite.org/about.html.

[TET+ay] Steven Tuecke, Doug Engert, Mary Thompson, Laura Pearlman, and Von Welch. Internet X.509 Public Key Infrastructure Proxy Certificate Profile. *Network Working Group*, Last accessed June 24, 2011. http://tools.ietf.org/html/rfc3820.

[TJÅAN09] Inger Anne T∅ndel, Martin Gilje Jaatun, and Åsmund Ahlmann Nyre. Security requirements for manets used in emergency and rescue operations. In *Security and Communication Networks (IWSCN), 2009 Proceedings of the 1st International Workshop on*, 2009.

[Ver11] Amandeep Verma. A study of performance comparisons of simulated ad hoc network routing protocols. 2011.

[VMC02] John Viega, Matt Messier, and Pravir Chandra. *Network Security with OpenSSL*. O'Reilly Media, 2002.

[WvLW09] Elias Weingartner, Hendrik vom Lehn, and Klaus Wehrle. A performance comparison of recent network simulators. In *Communications, 2009. ICC '09. IEEE International Conference on*, 2009.

[YLY+04] Hao Yang, Haiyun Luo, Fan Ye, Songwu Lu, and Lixia Zhang. Security in mobile ad hoc networks: Challenges and solutions. *Wireless Communications, IEEE*, 2004.

Appendix A

BATMAN Routing Protocol

A.1 Host Network Announcement (HNA)

A HNA message has a fixed size of 5 Bytes. A nodes appends one or more HNAs when it wants to announce a gateway to another network or host.

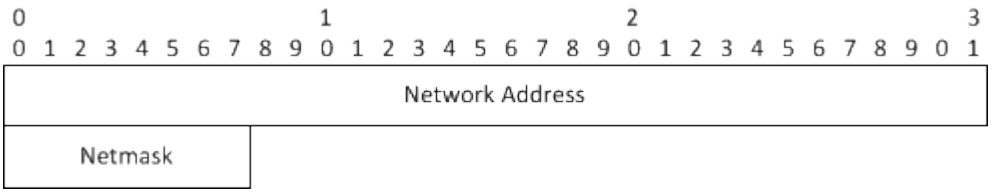


Figure A.1: HNA message Format.

The different fields found in this message are:

- **Netmask**
  Indicates the size of the announced network.
- **Network Address**
  The IPv4 network address of the announced network.

A.2 Sequence Numbers and Sliding Window

Sequence numbers cycle from 0 to $2^{16} - 1$ and start from 0 again when reaching the maximum value. Since the number range is limited, all arithmetical operations done must be performed using using modulo 2^{16}.

In-Window Sequence Numbers:
These numbers represents the latest accepted sequence numbers. The window ranges from the Current Sequence Number from this Originator to `WINDOW_SIZE` - 1 Sequence Numbers below it. The Current Sequence Number is not updated if an Originator Message (OGM) from this Originator is received with an In-Window Sequence Number, but it must however be memorized that an OGM with Sequence Number has been received.

Out-Of-Range Sequence Numbers:

These are all the Sequence Numbers that are not within the In-Window range and considered as the the new or next-expected Sequence Numbers. The Current Sequence Number is set to the Sequence Number in an received OGM if this number is out of range. Thus the sliding window is moved and the number which are not inside the window anymore are dropped.

Appendix B

BATMAN Ns-3 Module Details

The class inherits two virtual functions, `RouteOutput()` and `RouteInput()`, from the abstract base class `ns3::Ipv4RoutingProtocol`.

The `RouteOutput()` function is used by transport protocols and queries the routing cache for an existing route for an outbound packet. The Linux equivalent for this function is ip_route_output(). To route an inbound packet which is to be forwarded or locally delivered, is done by the `RouteInput()` function. This function is equivalent to the Linux ip_route_input() function.

Appendix C

Simulation Details

This appendix explains how the simulations with the network simulator ns-3 was conducted including details about the hardware.

C.1 Hardware Details

The simulations were performed on a computer with the following hardware specifications:

- Intel Core 2 CPU 6400 @ 2.13 GHz
- 4 GB memory

In addition, the computer was running Ubuntu 10.4 LTS - the Lucid Lynx.

C.2 ns-3 Simulation Details

The stable release at the time of writing, ns-3.10 version, was used when performing the simulations. A simulation script written in bash/sh compiles and runs the simulations scenario file with different parameter. The simulation scenario file store the data and statistics during the simulations to a SQLite database. After the simulations are done the script continues and queries the database to create data points which is then used to create graphs using Gnuplot.

The workflow done by the script is shown in Figure XX.

Appendix D

Simulation Scenario

Appendix E

Simulation Script

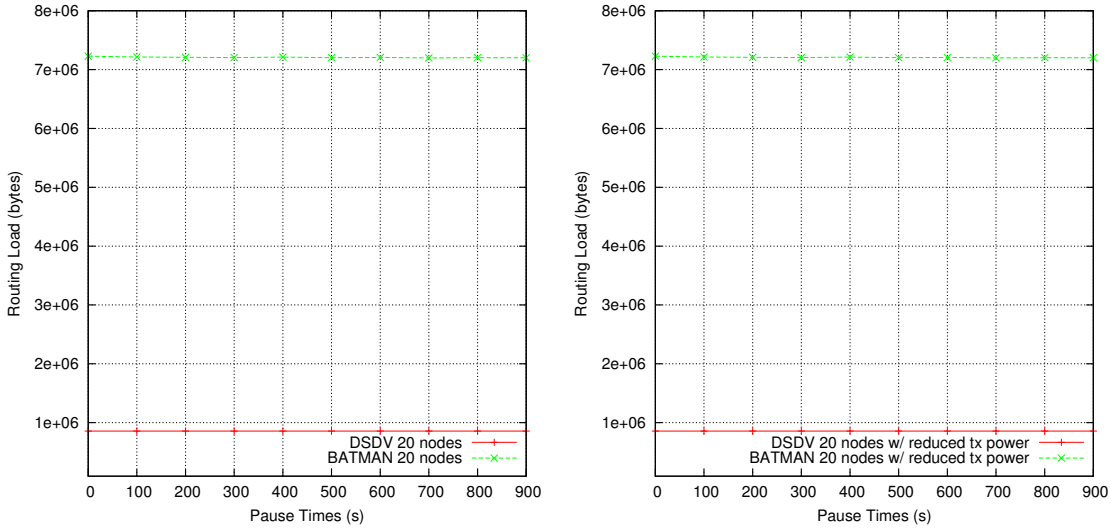
E.1 Script

```
#!/bin/sh

#PAUSE="0 200 400 600 800 900"
#TRIALS="1 2 3 4 5 6 7 8 9 10"

PAUSE="0 100 200 300 400 500 600 700 800 900"
TRIALS="1 2 3"

echo WiFi Experiment Example

pCheck=`which sqlite3`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sqlite3 (wifi-example-sim does not)
      ."
  exit 255
fi

pCheck=`which gnuplot`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires gnuplot (wifi-example-sim does not)
      ."
  exit 255
fi

pCheck=`which sed`
if [ -z "$pCheck" ]
then
  echo "ERROR: This script requires sed (wifi-example-sim does not)."
  exit 255
fi

export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:bin/

# Remove existing database
```

```
if [ -e data.db ]
then
  echo "Kill data.db? (y/n)"
  read ANS
  if [ "$ANS" = "yes" -o "$ANS" = "y" ]
  then
    echo Deleting database
    rm data.db
  fi
fi

# Compile the simulation scenario with the different parameters, and
    run the number of trials
for pause in $PAUSE
do
  for trial in $TRIALS
  do
    echo
    echo Pause time $pause, Trial $trial
    waf --run "annestest --format=db --batman=0 --seed=$trial --pause=
        $pause --ns3::YansWifiPhy::TxPowerStart=8.9048 --ns3::
        YansWifiPhy::TxPowerEnd=8.9048 --run=run-$pause"
  done
done



# Create SQL command
CMD="SELECT rx.run, avg(cast(rx.value as real)/cast(tx.value as real))
    FROM Singletons rx, Singletons tx
    WHERE rx.variable = 'rx' AND tx.variable='tx'
    GROUP BY rx.run
    ORDER BY rx.run ASC;"
#CMD="SELECT s.run, avg(s.value) FROM Singletons s GROUP BY abs(s.run);
    "

#mv ../../data.db .

# Query the SQLite Database
sqlite3 -noheader data.db "$CMD" > annestest.data

# Parse the data
sed -i "s/run-//" annestest.data
sed -i "s/|/   /" annestest.data

# Run gnuplot script and create graph
gnuplot dsdv_pdr.gnuplot

echo "Done; data stored in annestest.data, plot created, remember to
    check seed!"
```

Appendix F

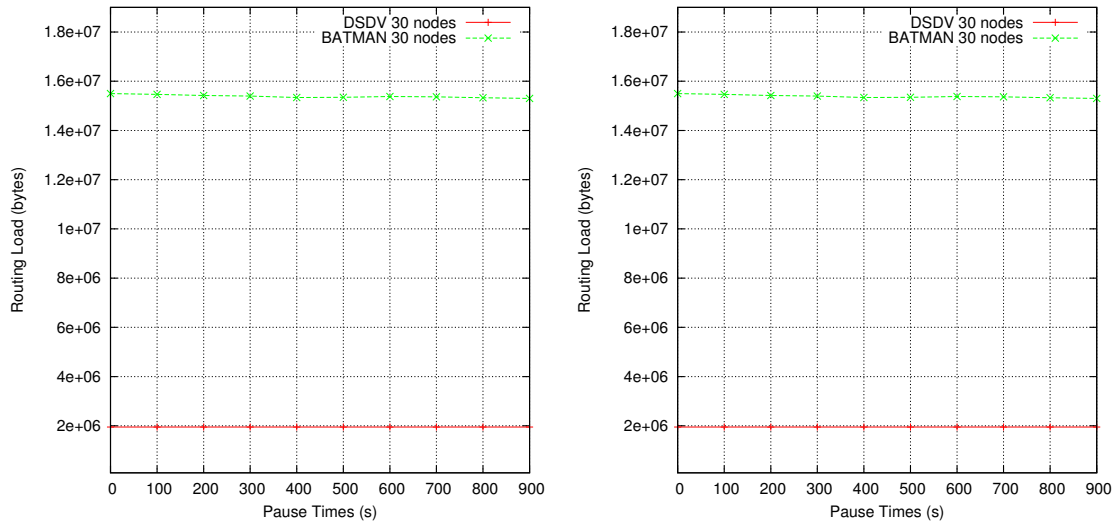
Additional Simulation Results



(a) Routing overhead with varying pause times.

(b) Routing overhead with varying pause times and reduced transmission power.

Figure F.1: Results from BATMAN and DSDV running with 20 nodes and 20 source and sink pairs where the transmission power has been reduced.



(a) Routing overhead with varying pause times. (b) Routing overhead with varying pause times and reduced transmission power.

Figure F.2: Results from BATMAN and DSDV running with 30 nodes and 30 source and sink pairs where the transmission power has been reduced.