A Communication Layer for Distributed Decisions Making

by

Paul Kaufmann

A thesis
presented to the
Rotterdam University
in fulfilment of the
thesis requirement for the degree of
Bachelor
in
Technical Computer Science

Spijkenisse, The Netherlands, 2004

Abstract

This report represents a research project and implementation project called ‘A Communication Layer for Distributed Decision Making’, which was initiated in January 2004 at the Delft University of Technology in the Netherlands. It aims at how messages are being routed from a source to a destination in a wireless ad hoc network. Project supervisor is professor dr. drs. L.J.M. Rothkrantz and Paul Kaufmann, student at the University of Rotterdam, carries out research and implementation work.

Ten years ago PDA’s were introduced. With this technology it is possible to communicate directly to other PDA’s instead of using a centralized network. It can be applied in different professions for example in medical emergency, police and fire fighters etc. But there is not yet a optimal routing algorithm to distribute information to other PDA’s.

This thesis contains a result of a research of existing routing algorithms and an implementation of a routing algorithm in an ad hoc network. Different routing algorithms have been investigated such as ARA, DSR and ZRP. ARA proved to be the most promising in comparison with other routing protocols, but some adjustments have been made to ensure that data has arrived at the destination. However ARA is not superior to other routing protocols.

An implementation has been made to see what will happen if this ARA routing algorithm is applied in the real life. The design and implementation can be found in this thesis. Eventually the results of the simulation will be presented.

Keywords: routing algorithms, ad hoc networks, ARA, DSR, dynamic source routing, ant based routing algorithm

Acknowledgements

I would like to thank the gentlemen, dr. drs. L.J.M. Rothkrantz, ir. M. Abdelghany and ir. J.P. Manni for the challenging assignment and inspiration.

Furthermore I would like to thank B. Tatomir BSc. for helping me by giving me an explanation of the city program.

In general I would like to thank everyone who helped me (in)directly with the project. The help mainly worked motivated, but worked inspiring and contributed the thesis.

With this note I want thank everyone for the support.

Thank you.

Paul Kaufmann

Table of Contents

List of Figures

List of Tables

Chapter 1 : Introduction

1.1 Introduction

September 11th, hijacked airplanes hit the World Trade Center in New York and the Pentagon. Hundreds of policemen and firemen were sent out to the scene. Emergency 911 was overwhelmed by calls. Rescuers were forced to take rapid-fire and life-death decisions based on incomplete information, which lead to a bigger death toll. Building 2 was said to be secure and there was no need to evacuate building 2. After the attack, normal communication traffic was even doubled, because three of the major broadcast networks were located on the top of the North World Trade Tower, communication coverage was limited.

1.2 Problem definition

At the moment new networks are being introduced that don't use a centralized point, which makes it possible to apply in the reality. But before this can be accomplished communication agreements must be made.

The problem definition, how can information efficiently be distributed without using a centralized point, can be divided into several sub problems:

- How to create a route?
- What will be the structure of the message?
- How will the message be processed?
- How is information being distributed over the network?

1.3 Motivation

For some disasters or accidents there are no standard procedures and could lead to confusing situations. By providing new information of disasters or accidents, probably more lives can be saved and the situation can be restored faster, which decreases the material damage in the area.

1.4 Project description

There are two possibilities to investigate this problem, by making a simulation in reality or making a simulation program. A simulation program has the advantage that more traffic can be simulated and in reality this situation will be difficult, because information must finally be collected at one point. That is why a simulation program will be made.

1.5 Background

The project will take place at the Technical University of Delft (TU-Delft). The mediamatica department is one of the many departments and is part of the Electronic, Mathematics and Informatics Faculty (EEMCS).

To simulate a disaster different features must be build, like how to exchange information, what to do in a certain situation and how to determine a route. Because this takes a lot of work, there has been chosen for an approach to divide the functionality in different (sub)projects, also called layers. The main project 'Crisis Management' is divided into different layers; the Agent Layer, the Traffic Simulation Layer and the Communication Layer, see figure 1-1.

The *Traffic Simulation Layer* also named 'Dynamic vehicle routing using ant based control' determine the route. In this project a program has been made by Ronald Kroon, which can simulate traffic [DYAB]. The traffic is moving from a random source to a random destination. The used routing information is not based on the shortest distance, like the ANWB works, but based on the shortest time by means of dynamic information. The program is improved and extended by B. Tatomir BSc [BOT].

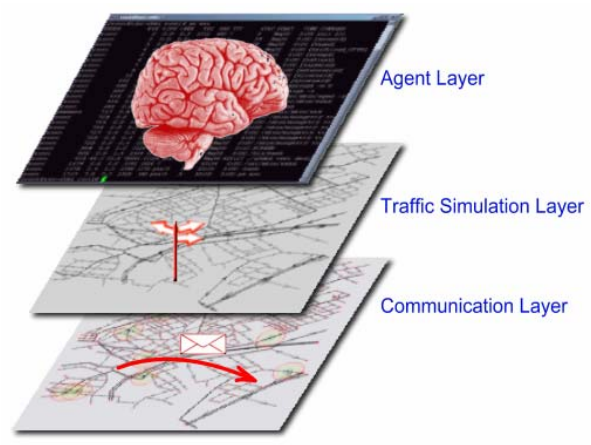


Figure 1-1 Layers

At the same time when the Communication Layer functionality is build, the *Agent Layer* project will be started. The main objective of that project will examine and implement the possibilities of adding agents in a network. The agents will watch the current situation of the environment. If for example smoke is detected then the information will be passed to other nodes in the surrounding area. Based on the information it takes actions. Martin Bethlehem executes the project.

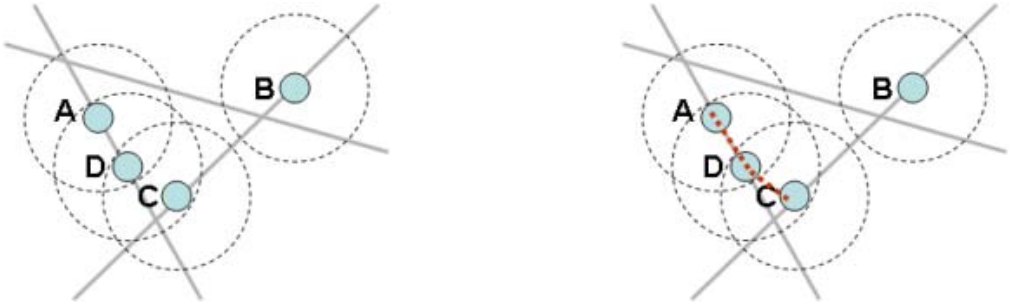


Figure 1-2 Ad Hoc Communication

There are different devices that can communicate with each other without using a centralized point. In the simulation program PDA devices will be used. If PDA's are within each other's range, messages can then be exchanged like A and D in figure 1-2. Because C is connected to D, PDA A can also communicate with C. On top of the traffic map there is also a *Communication Layer* in which communication connections are visible. The Communication Layer or the project 'A Communication Layer for Distributed Decisions Making' will be described in this paper.

1.6 Approach

The project, named 'A Communication Layer for Distributed Decisions Making', is divided into five phases; preparation, research, design, implementation and the final phase closing the project. The first phase preparation as the name indicates will occupy everything related with the assignment of the project, for example is the main aim possible and realizable. Before the research phase can be started a plan of approach must be finished. In the research phase all information, that is necessary to accomplice the objective, will be investigated. The third phase design produces a first design of the program. The implementation phase will use this design to produce the program. The program will be tested for errors. The final phase will be used to close the project. This phase produces a presentation of the project and documentation of the program that has been made.

1.7 Delimitations / Assumptions

There are a few assumptions such as that the communication range of the PDA will have a shape of a circle and that a maximum range always can be reached and the communication will use the standard 802.11b.

The program will not be implemented on a PDA, but on a pc. It is taken into account that a maximum number of 600 PDA's can be simulated. The battery usage will not be implemented, but memory usage and hard disk usage is realized.

Chapter 2 : Theoretical Background

2.1 Theoretical Background

In this chapter, an explanation will be given concerning definitions, which will later be used in the thesis.

2.2 Personal Digital Assistant (PDA)

A PDA is an electronic device that organizes information. Basically it is a small computer. It is also known as Handheld or Palmtop. Most PDA's have some basic features like a Day planner, Memo, Calculator, To-Do-List, Address book, Email, on screen keyboard and handwrite recognizer.

Big differences between normal computers and PDA's are that the system components of a PDA are less powerful. Disk space, memory, processor speed is less than a normal computer and the PDA uses a battery that must be recharged after some time. But PDA's can be taken anywhere and most of them have the possibility to easily connect to another network or device.

2.3 Hybrid / Peer to Peer

Peer to Peer network is a network in which devices or stations directly communicate with each other without using an access point see figure 2-2. An access point is a hardware device or computer software that acts as a communication hub for users of a wireless device to connect to a wired LAN. Peer to peer is also referred as Ad-Hoc. Wireless Fidelity (Wi-Fi) is an example, which uses the ad-hoc method. A hybrid network is the opposite of a peer to peer network, it uses a centralized access point like figure 2-1. If this access point is down, communication is not possible anymore. Examples of hybrid networks are GSM (Global System for Mobile Communications) and UMTS (Universal Mobile Telecommunication System).

Figure 2-1 Hybrid

Figure 2-2 Peer to Peer

2.4 Mobile Ad Hoc Network (MANET)

A mobile ad hoc network, also called MANET, is a collection of autonomous nodes that can communicate with each other in a decentralized way. A node can be a computer or some other device, in this case it are PDA's. Every node has got a unique network address. The network topology is dynamic, new nodes can arrive in the network and other nodes can leave the network.

Because nodes within the network are communicating wireless, interference, fading and noise must be taken in account. The MANET bandwidth is less than other networks such as the Internet. In a MANET a node cannot directly notice if a node that was in range still is in range. The Internet on the other hand can notice if a connection still exists. A big problem with MANET's is the security, because attacks are different compared to Internet attacks.

Potential Applications:

1. Conferences/Meetings
2. Search and Rescue
3. Disaster Recovery
4. Automated Battlefields

2.5 Ant Based Control Algorithm

Ant based control algorithm, as the name may indicate, is based on the ant swarm intelligence. This algorithm is based on the behaviour of ants searching for food. An ant chooses every time it encountered a crossing which direction it will go to. The ant leaves a smell, called pheromone, on the route it's been on, which other ants can smell. After a certain time the intensity of the pheromone concentration will decrease.

An example: somewhere is food located. The ants take different routes to the food. When an ant has reached the food other ants can smell the concentration of pheromone. Because the ant that arrives first at the food has left a route with a higher concentration towards the food, more ants will eventually choose this route. The pheromone on the shortest route to the food will also be incremented because more ants walk on this route.

This algorithm can be used to find the shortest route in time between two points. It can be applied in many applications; artificial intelligence, networks or medical researches.

Chapter 3 : Literature study

3.1 Literature study

This chapter will give all information that has been researched and is related to the subject of the project.

3.2 Programming Languages

A few aspects were important for the decision which program language to choose. These aspects were the execution speed, the features, future development and how fast development was possible. Java was no option because it did not satisfy to a fast runtime and missed some features [CVERSJ]; it was much slower than other programming languages such as C-sharp(C#), J-sharp(J#), Delphi and C++. Delphi has a very high execution time and the latest version of Delphi enabled the possibility of deploying programs in the .NET framework. However Delphi doesn't use a garbage collection this must be manually taken care of.

Dot Net allows different program languages exchange information, this allow programs to be developed in different program languages. J# still has all the benefits of the Java programming language as a simplified object-oriented programming language (no pointers, and so on) and it's based on JDK 1.1.4. However J# is more to help Java programmers make the switch then anything else.

Two programs written in C# and Java have been tested for the performance of C#. The first program to test the graphical performance and the second to test the runtime performance. It showed that C# provided a better performance than Java [ETJC].

3.2.1 Programming editor

Visual Studio .NET provides an editor and drop and drag window design mode, instead of writing the window graphical look it can easily be modified by drawing the components with the mouse. Visual Studio .NET 2003 provides developers with tools for designing and building distributed applications for Microsoft Windows, the Web, and mobile devices.

Features of Visual Studio .NET 2003:

- Graphical applications, databases, and business processes.
- Functionality and architecture of an application can be defined in XML Web services.
- Application can be deployed faster by using templates and share file possibilities.

3.2.2 Conclusion

C# has been chosen for the programming language, because it provides a rapid development with Visual Studio .NET and many resources are available and the performance is good. C# also offers more elegance and simplicity over C++ and proved to be quite fast. Other languages of the .NET framework can combine or use the functionality of the program, which can be written in another language.

3.3 Routing algorithms

A major issue with ad-hoc networks is to find the optimal routing algorithm for Mobile Ad-hoc Networks since nodes are constantly moving and can be disconnected at any time. There are many routing protocols such as Dynamic Source Routing (DSR), Destination-Sequenced Distance-Vector Routing (DSDV), Contention-Based Forwarding (CBF), the Ant-Colony Based Routing Algorithm (ARA), the Ad-Hoc On-demand Distance Vector (AODV), Temporally-Ordered Routing Algorithm (TORA), Zone Routing Protocol (ZRP) and many more.

All routing protocols use different approaches:

- Proactive; check after a while which routes still exists, DSDV
- Reactive; calculates the route on demand, DSR, AODV, TORA
- Hybrid; creating a centralized point, in the ad hoc network

Even today no perfect routing algorithm exists, some algorithms use less routing data but cost more network traffic. That is why this section will only discuss the mostly commonly used protocols and the technique involved. Finally a conclusion will be given.

3.3.1 Dynamic Source Routing (DSR)

This protocol as the name may indicate is source based [DSR]. The network is self-organizing and self-configuring which require no administration using DSR. A major advantage is that DSR rapidly adjust to a fast changing network.

If a message has been send, each neighbour rebroadcasts this Route Request if the destination has not been found in the route cache, it will add its own address in the header of the packet (see figure 3-1). Each node on the route will confirm if the data packet successfully has been received. If a node finds it's own address already listed, it discards the Request. When the Route Request is received by the destination a Route Reply will be returned to the sender.

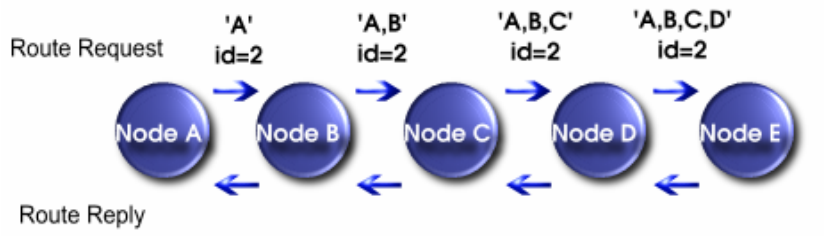


Figure 3-1 Routing example of DSR

The sender has a Sent Buffer that contains a copy of the message that has been send. If a Route Reply is received the message can be removed from the buffer.

While this process could use up a lot of bandwidth, DSR gives each node a route cache to reduce the number of messages to be sent. It will listen to other messages for additional routing data to add to the cache. When a node receives a Route Request it will first examine its route

cache for the received destination. If found in the route cache it will use this route instead of forwarding the Route Request to it's neighbours.

DSR has the advantage that no routing tables are required, since the entire route is included in the packet header of the message. Route caching can significantly reduce the number of control messages being sent.

Two primary disadvantages of DSR are:

- DSR is not scalable to large networks; the diameter of the network cannot be greater than 10 hops
- DSR requires more processing resources than most other protocols; each node must spend much time in processing any control messages

3.3.2 Contention-Based Forwarding (CBF)

The general idea of position-based routing is to select the next hop based on position information such that the packet is forwarded in the geographical direction of the destination [CBF]. An important characteristic of position based routing is that forwarding decisions are based on local knowledge. Another characteristic of position based routing is that it is not necessary to create and maintain a global route from the sender to the destination. This is why position-based routing is commonly regarded as highly scalable and very robust against frequent topological changes.

The forwarding decision is based on the node's own geographical position, the position of all neighbours within transmission range and the geographical position of the destination.

To send a message the sender requests the position of the destination and then includes it in the header of the packet. Given this information, the node forwards the packet to one of its neighbours such that the packet makes progress toward the destination. All neighbours who received the packet will compete with each other. In this contention period, the neighbours determine how well they are suited as the next hop for the packet. The node that wins suppresses the other neighbours and establishes itself as the next forwarding node.

There are several suppression methods:

- Timer-based contention: a timer is set (the progress towards the destination) and when the timer expires the corresponding node responds. The timers of all other nodes are cancelled and their responses suppressed.
- Basic Suppression: if a timer at a node expires it assumes that it is the next hop and broadcasts the packet. When other nodes receive this broadcast and still have a timer running for the packet, it will be cancelled. It is possible that the broadcast does not reach all the other nodes and that will cause packet duplication.

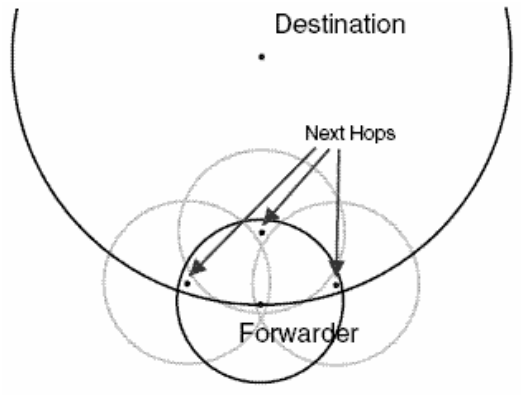


Figure 3-2 CBF Routing

- Area based suppression: the area from which the next hop is selected is artificially reduced. The idea is that all nodes are within transmission range. Area-based eliminates duplications of packets caused by nodes that are not within transmission range of each other, but packet duplication caused by the timer is not prevented (see figure 3-3 on the next page).
- Active Selection: the forwarding node broadcasts a control packet, RFC (request to forward). Every neighbour that receives the RTF checks if it provides forward progress. If so the node sets a reply timer like the basic suppression scheme. When the timer expires a control packet CTF (clear to forward) is send to the forwarding node. A node that receives the CTF packet deletes its timer and is suppressed. The forwarding node receives the CTF packets and selects the node with the largest progress and transmits the packet to this node.

CBF does not use beacons and does not keep information of neighbours in its table for forwarding, but to make sure that there are no duplicated messages it will cost more bandwidth. A disadvantage is that CBF have to be well tuned for optimal performance and the direction towards the destination is not always the best route.

3.3.3 Zone Routing Protocol (ZRP)

The separation of a nodes local neighbourhood from the global topology of the entire network allows for applying different approaches - and thus taking advantage of each technique's features for a given situation. These local neighbourhoods are called zones; each node may be within multiple overlapping zones, and each zone may be of a different size [ZRP].

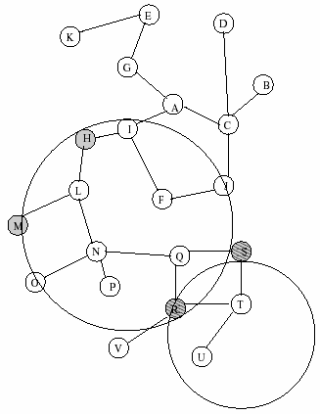


Figure 3-3 Zones in a MANET

Every node periodically broadcasts discovery messages in order to keep a map of it's neighbours up-to-date. There are two protocols the Intrazone Routing Protocol (IARP) and Interzone Routing Protocol (IERP). The IARP is responsible for the communication with the peripheral nodes and is commonly a pro-active protocol. The IERP provides communication with the peripheral nodes only. Peripheral nodes are nodes whose minimum distance to the node in question is equal exactly to he zone radius (see figure 3-3). The other nodes are referred as interior nodes. Each node can have its own zone, so zones can heavily overlap.

To send data the source first checks if the destination is within its zone. If the destination is in its zone the destination is known, because of the broadcasts it sends regularly. If the destination node is not within its zone, the node sends a request to all its peripheral nodes. The peripheral nodes will also check if the destination node is within its zone and will forward the data to its peripheral nodes if the destination is not within its zone. This action will be repeated till the destination is found within its zone then the route will be replied.

ZRP provides a hybrid approach in contrast with other MANET's protocols. Also ZRP is more suitable for larger networks. A disadvantage is that the network traffic is very intensive because of the hybrid architecture.

3.3.4 The Ant-Colony Based Routing Algorithm (ARA)

This routing algorithm is based on the idea of the behaviour from the food searching of ants [ARA]. While an ant walks it deposits pheromone (kind of cumarin, which ants can smell). Over time the insensitivity of pheromone will decrease. Suppose two ants are leaving from the nest at the same time searching for food.

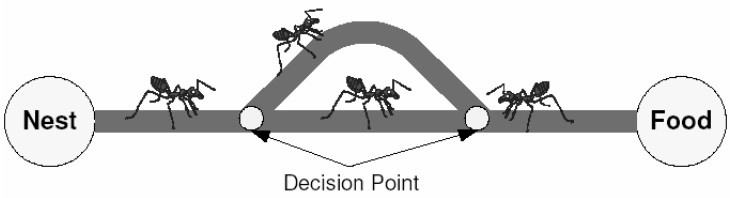


Figure 3-4 Ant Based Route Searching

They take two different paths; one path is shorter than the other path. So the ant that arrives first has taken the shortest path to the food. After a while the shortest path will have a higher concentration of pheromone than the other path. Because the ants are using the shortest path to the nest they will eventually all choose for the shortest path (figure 3-4).

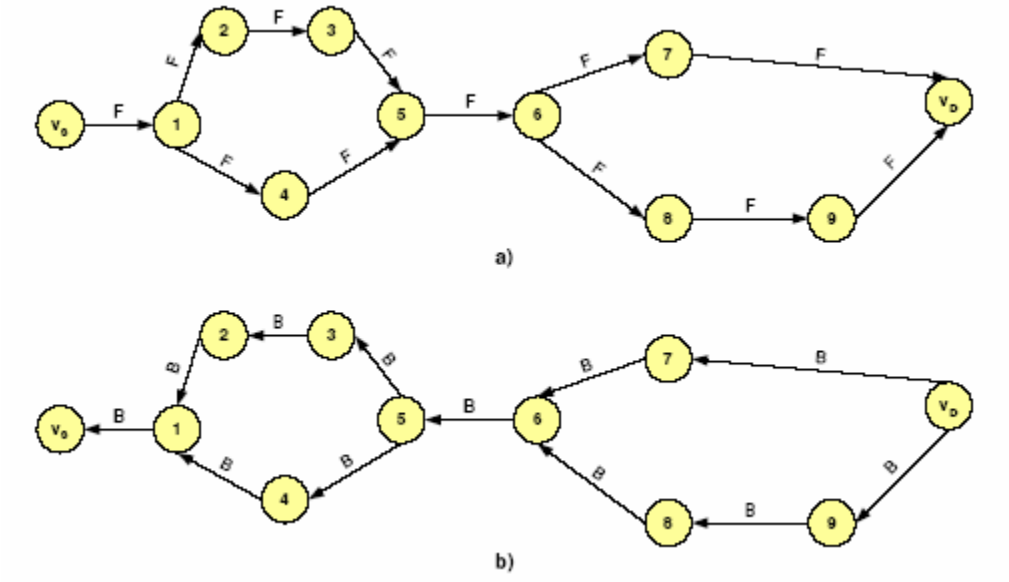


Figure 3-5 Route discovery

To create a new route a forward ant (FANT) is initiated at the source and a backward ant (BANT) at the destination. When a node receives a FANT for the first time it will add a record in the routing table (destination address, next hop, pheromone value). The node interprets the source address of the FANT as destination address, the address of the previous node as the next hop and computes the pheromone value depending on the number of hops it took the FANT to reach the node. The node forwards the FANT to its neighbours. If the FANT reaches the destination, a BANT will be returned to the source. The BANT will use the take the same course of action as the FANT. Once a path is established regular data packets are essential to maintain the path (see figure 3-5).

A high decay rate will quickly reduce the pheromone value. If a regular time interval of one second is taken the decreasing formula is $(1-q)=0.1$. Every time data passed by the pheromone value in the routing table will be increased by 0.1.

Duplicated FANT's can be identified through the unique sequence number and are removed. If a node receives a duplicated packet, it will set the DUPLICATE_ERROR flag and send the packet back to the previous node. The previous node deactivates the link to this node, packets cannot be sent anymore to this direction. The ROUTE_ERROR is set when the node misses an acknowledgment on the MAC layer. When this occurs first the link will be disabled and then it will check if there exists another route to the destination node. If no route exists to the destination it will inform its neighbours, hoping that they can forward the packet towards the destination. If none of the neighbours can forward the packet towards the destination the packet is send back to the previous node until the source has been reached.

Control bandwidth overhead is minimized and remains constant across several degrees of the network volatility.

3.3.5 Conclusion

There are many routing protocols, but none of them provides the perfect solution. However some routing protocols offer techniques that are more effective than other routing protocols. In this section different techniques have been discussed such as ant based, source based and position based.

Position based is not a real solution because when moving towards the destination is not always the correct path. Besides, does the forward node provide a good transmission speed? Source based costs more bandwidth than other protocols, but the message will arrive at the destination if there is an existing path. However the source-based networks are very small and cannot be used in large-scale networks. The ant-based protocol requires less bandwidth than the mentioned routing protocols. But a TimeToLive must be added, because we do not want the message travelling an entire large network. This TimeToLive field will be decreased every time it passes a node until it reaches zero then it will be discarded. Before the ant-based protocol can be implemented, the routing protocol must be extended or adjusted to correctly work in a MANET.

Which routing protocol must be taken for the implementation of a program depends on the purpose of the program. For example, if many services are offered and the network traffic must be minimized, perhaps local broadcasts or spreading services can be a solution. Finally, when one or more techniques are chosen the techniques can be combined, tuned or extended.

For an optimized routing protocol, every existing technique must be examined and adjusted for the purpose for a better performance.

3.4 IEEE 802.11b

The purpose of this section is to give an idea of how the IEEE 802.11b standard works. This standard is necessary for the program to make the simulation more realistic. Not every detail of IEEE 802.11b will be discussed, only the global idea.

3.4.1 Introduction

To ensure the compatibility and reliability among all Wireless LAN's (WLAN) devices, a standard is necessary. The institute of Electrical and Electronics Engineers (IEEE) has just provided that. The first standard was IEEE 802.11 that was defined in 199, followed by IEEE 802.11a and finally IEEE 802.11b in 1999. IEEE802.11b is an improvement of IEEE802.11 to support higher data transmissions of up to 11 Mbps instead of 1/2 Mbps.

802.11b has a range of about 50 metres. 802.11b has a maximum throughput of 11 Mbit/s. However a great percentage is used for the overhead of communication data. The throughput is scaled to 11 Mbit/s, 5.5, 2 and finally 1. This depends on the signal strength. Metal, water, and particularly thick walls absorb 802.11b signals and decrease the range drastically. The protocol can also operate in fixed point-to-point situations, in which significant ranges can be reached.

In order to increase the speed to 22, 33 and 44 Mbit/s extensions of the IEEE 802.11b have been made. But the IEEE does not support the extensions. Many companies call enhanced versions "802.11b+".

3.4.2 Architecture

IEEE 802.11 divides the system into cells, where each cell is called Basic Service Set. A Base Station, also called Access Point (AP), controls the Basic Service Set (BSS).

Figure 3-6 An Access Point

Most Access Points are connected through some kind of backbone, called Distributed System (DS).

3.4.3 Avoiding Collisions

Consider a node that has no idea of its neighbours. Another node has the same problem. If they both send a message at the same time it will result in collisions. The solution for this problem is Carrier Sense Multiple Access with Collision Avoidance or CSMA/CA. CSMA/CA avoids this problem by listening first and then sends a short message (figure 3-7).

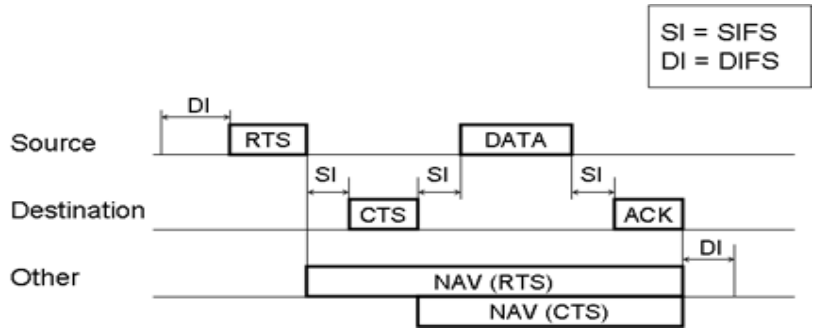


Figure 3-7 Carrier Sense Multiple with Collision Avoidance

A node that wants to transmit must be free for a specified time, called Distributed Inter Frame Space (DIFS). If allowed to transmit, a short control packet called the Ready To Send (RTS) will be send. The RTS control packet contains the destination address and the duration of the transmission. Other nodes now known that they must wait before they can transmit, called the Network Allocation Vector (NAV). If the destination is free, the destination will respond after a given time, the Shorter Inter Frame Space (SIFS), with a control packet called Clear To Send (CTS). The SIFS is a fixed value that allows the transmitting node to switch back to receive mode. The source can now send a message back without collisions. Each packet is acknowledged by the destination. The packet will be retransmitted if no acknowledge is received. This is called the 4-way handshake (see figure 3-8). More information of IEEE frames can be found in the Appendix.

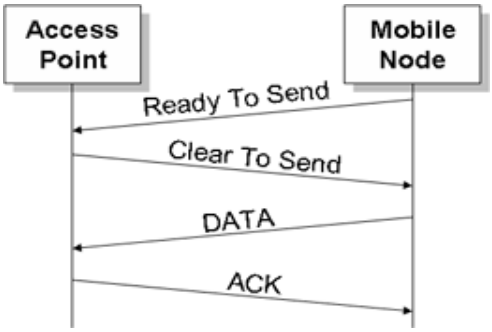


Figure 3-8 4-way handshake

3.5 Neighbour Searching

3.5.1 Introduction

The Quad Tree method is tree based neighbour searching algorithm. It divides an area into four equal squares, each of those squares are also being divided until a certain level has been reached.

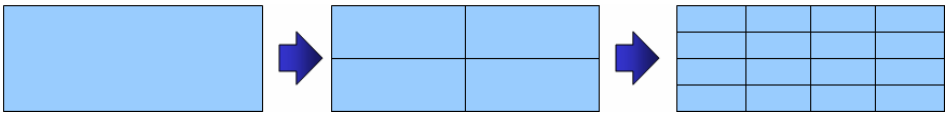


Figure 3-9 Dividing a Square

The example in figure 3-9 can be transformed into a tree like figure 3-10. There are different approaches of implementing a quad tree. The first approach will link nodes to other nodes. The second approach use levels. First an explanation of a tree walk will be given.

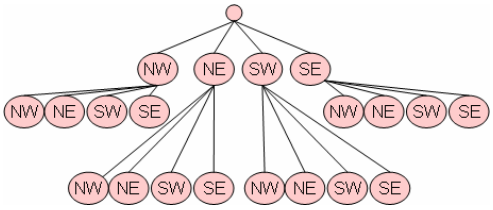


Figure 3-10 A Quad tree

3.5.2 Walk Through Approach

Suppose a point such as in figure 3-11 (the black dot in the square) must be placed in the quad tree. Because the point fits in the area of the root/upper node, the areas of node on the second level will be checked for the placement of the point. This procedure will be repeated until the lower level has been reached or if the point does not fit in the area. A node then adds this point to its container. In this example 3-11 a point will walk through the upper left area, the North West node and finally the point will be placed in the North East node.

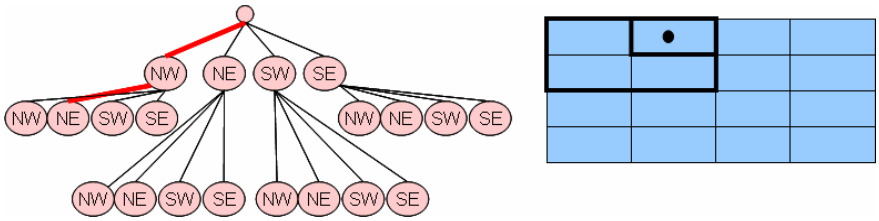


Figure 3-11 Searching in a Quad Tree

To check if the point fits in the area of a node, first the centre from the area is taken from the node. Secondly this centre point is compared to the point that must be placed. Is the point located in the upper left corner, it will be placed in the NW node/TreeNode.

In the previous example a point was given, however placing an area in the Quad Tree is also possible. The same actions must be taken, however there are some differences. In the simulation program, PDA's will have a range. This range has the shape of a circle, but the Quad Tree requires a square. Instead of checking precisely a point on the circle only the corners of the square will be used to check if a PDA is in range of an area, see figure 3-12.

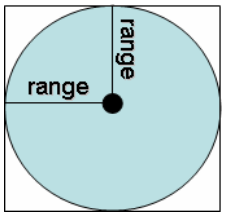


Figure 3-12 Checking ranges

Checking if a node is in an area can be accomplished by comparing the upper-left corner and bottom-right of square with the upper-left corner and bottom-right corner of the area. The upper-left corner of the PDA square must be greater or equal to the upper-left corner of the area. The bottom-right corner of the PDA square must be less or equal to the bottom-right corner of the area. The procedure in code can be found in the table 3-1.

```
if ((node.m_area.X <= pda.Location.X + pda.TransmissionRange &&
       node.m_area.Y <= pda.Location.Y + pda.TransmissionRange) ||
   (node.m_area.Right >= pda.Location.X - pda.TransmissionRange &&
       node.m_area.Bottom >= pda.Location.Y - pda.TransmissionRange))
      return true;
else
      return false;
```

Figure 3-1 Searching a PDA

When searching for neighbours first the node where the PDA is located in must be found. Once the node has been found all the nodes beneath must check the PDA's for their range. Checking if a PDA is within range can be calculated by the Pythagoras formula ($distance = \sqrt{(\Delta x)^2 + (\Delta y)^2}$). The nodes above the founded node, that contains the PDA, must also be checked if those PDA's are within range. Finally a list of neighbours can be reported.

3.5.3 XOR Approach

The second method is a different approach; instead of linking nodes to other nodes it will directly place a point in the quad tree. But the XOR method also uses a square like the walk through method as a replacement for the circle shape.

Using XOR, the level can be calculated where the area must be placed. Two calculations have to be made, one for the x and one for the y. Before the upper-left x XOR bottom-right x can be compared with upper-left y XOR bottom-right y, the highest bit must be taken. This means that a bit value for example 100110, 38 decimal, must be transformed to bit value 100000, 32 decimal. The maximum value of the comparison can be used to calculate the level. If the value is 256 then the level is 0, is the value 128 then the level is 1 etc. The final level is level 8.

Now that the level is known, the PDA must be placed at the correct node. Suppose every level of the Quad Tree has got a two dimensional array/matrix of nodes see figure 3-13, it can directly be placed by a simple formula see figure 3-14 [indexX][indexY].

	0	1	2	3
0	[0][0]	[1][0]	[2][0]	[3][0]
1	[0][1]	[1][1]	[2][1]	[3][1]
2	[0][2]	[1][2]	[2][2]	[3][2]
3	[0][3]	[1][3]	[2][3]	[3][3]

Figure 3-13 XOR Approach

$$\frac{(pda.location.x - area.topleft.x)}{width} = indexX$$

$$\frac{(pda.location.y - area.topleft.y)}{height} = indexY$$

Figure 3-14 Place a point in a 2D Matrix

An advantage of this approach is that it is not necessary to check if the PDA is out of range. But a disadvantage of this approach is that it requires that the area has got a size of 2^n and still will use the walk through approach when searching for neighbours.

3.5.4 Quad Tree Implementation

The design looks could look like figure 3-15, but of course you can implement your own design.

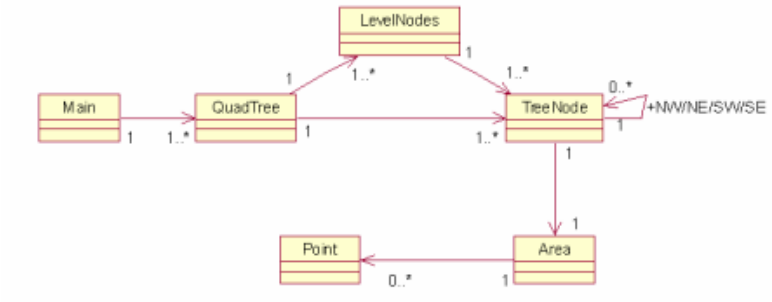


Figure 3-15 UML Design of a Quad Tree Implementation

Before the source code will be shown some explanation of the design will be given. The main class is the class that can be used to run the program. The TreeNode class has a connection to its own class these are the NW/NE/SW and SE areas.

3.5.5 Sector

This is a neighbour searching algorithm that is created by myself. This algorithm has been given the name sector because it divides the total area into smaller areas with all the same height and width. Every small area, a sector, holds a collection of nodes. Nodes can easily find neighbours, because they only have to search through the collection nodes of their neighbour sectors. However the sector algorithm has a disadvantage, it cannot be used when nodes have different ranges.

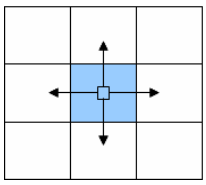


Figure 3-16 Sector Neighbour Searching

The entire area is divided into a two-dimensional array/ matrix. The width and height equals the range multiplied with 2½. The half is added to allow nodes to fit sometimes perfectly in the square; so all neighbours are located in the square if the PDA fits in the square. In table 3-2 the code has been given how a neighbour is being searched. If neighbours are being searched it will check if the PDA can reach other squares by taking the current location and adding or minus the range. For each edge of the square, where the PDA is located in, it will check if it can crosses the line (see figure 3-16). If the left line and the upper line can be crossed of the square, the square on the upper-left of the square will automatically be added. Instead of checking eight neighbour squares, it will only check the necessary neighbours that contain the neighbour PDA's.

```
bool xl = false, xr = false;
if (x-1 >= 0 && pda.Location.X - pda.TransmissionRange <
pda.TreeNode.m_area.X)  {
      // located on the left side of the rectangle
      nlist.Add(m_treeNodesArray[x-1,y]);
      xl = true;
} else if (x+1<sizearrayx && pda.Location.X + pda.TransmissionRange >
pda.TreeNode.m_area.Right) {
      // located on the right side of the rectangle
      nlist.Add(m_treeNodesArray[x+1,y]);
      xr = true;
}
if (y+1 < sizearrayy && pda.Location.Y + 10 >
pda.TreeNode.m_area.Bottom) {
      // located on the lower side of the rectangle
      nlist.Add(m_treeNodesArray[x,y+1]);
      if (xl)
            nlist.Add(m_treeNodesArray[x-1,y+1]);
      if (xr)
            nlist.Add(m_treeNodesArray[x+1,y+1]);
} else if (y-1 >=0 && pda.Location.Y - 10 < pda.TreeNode.m_area.Y) {
      // located on the upper side of the rectangle
      nlist.Add(m_treeNodesArray[x,y-1]);
      if (xl)
            nlist.Add(m_treeNodesArray[x-1,y-1]);
      if (xr)
            nlist.Add(m_treeNodesArray[x+1,y-1]);
}

ArrayList neighbours = new ArrayList();

for (int i=0;i<nlist.Count;i++) {
      TreeNode tmp = (TreeNode)nlist[i];
      for (int u=0;u<tmp.m_elements.Count;u++) {
```

```
            PDA pdatmp = ((PDA)(tmp.m_elements[u]));
            float distance = Distance( pdatmp.Location, pda.Location);
            if (distance <= pda.TransmissionRange && pda != pdatmp)
                  neighbours.Add(new Neighbour(pdatmp,distance));
      }
}
return neighbours;
```

Figure 3-2 Sector Neighbour Searching code

3.5.6 Sector Implementation

An implementation can be found in the Appendix …. It contains only one class named Sector. Sector will store all the small squares in memory by using a two dimensional array. Each element of the array contains a container of PDA's.

3.6 Conclusion

There are many neighbour searching protocols, but only a few of them provide a high performance on updating each time see figure 3-17. Because the simulation program needs to update all locations each time, static approaches that need to rebuild the data structure will not be an option. The best performance is the sector algorithm an implementation and performance can be found in the Appendix.

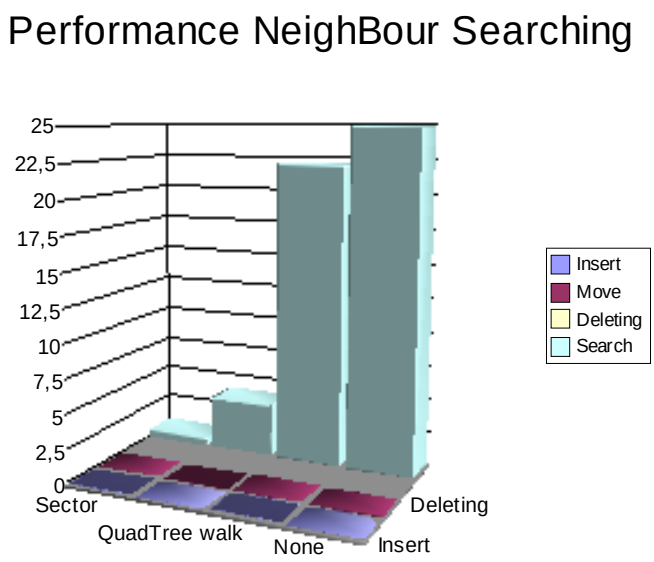


Figure 3-17 Performance Neighbour Searching

Chapter 4 : Design

4.1 Design Goals

It is essential to create a design before implementing the program. Because if the structure is not clear before implementing the program, the overview can be gone and errors may occur, that probably could have been avoided. This could lead to major problems in the development. Especially for complex or enormous programs it is necessary to give each component responsibilities. This ensures the flexibility of the software and adjustment can easily be made.

The Unified Modelling Language (UML) has got different diagrams and models. The class diagram has been chosen to reflect the structure of the program that must be developed.

4.2 Design

The ‘Ad Hoc Visualization’ as the name may indicate takes care of the visualization and will be referred as AHV. The ‘Ad Hoc Simulation’ simulates the environment and uses the protocol, which determines where the data must go to, it will be referred as AHS. These two programs communicate with each other by using a socket. Because the programs are using a socket communication it is possible to place the programs on different computers. It is also allows other programs communicate with this program by a socket connection. It even can replace the AHV program, if the server functionality is implemented. The implementation section will discuss how this can be achieved.

The AHS contains components, which hold files with specialist functionality. These components can be replaced by other components. For example if a different routing protocol must be simulated only a component have to be made and this component can replace the implemented version.

First the components of the AHS will be commented then the components of the AHV will be discussed. Both designs use an AEP, which is an abbreviation for a Application Entry Point. When the programs are started AEP will be called first and starts the program.

4.2.1 AHS

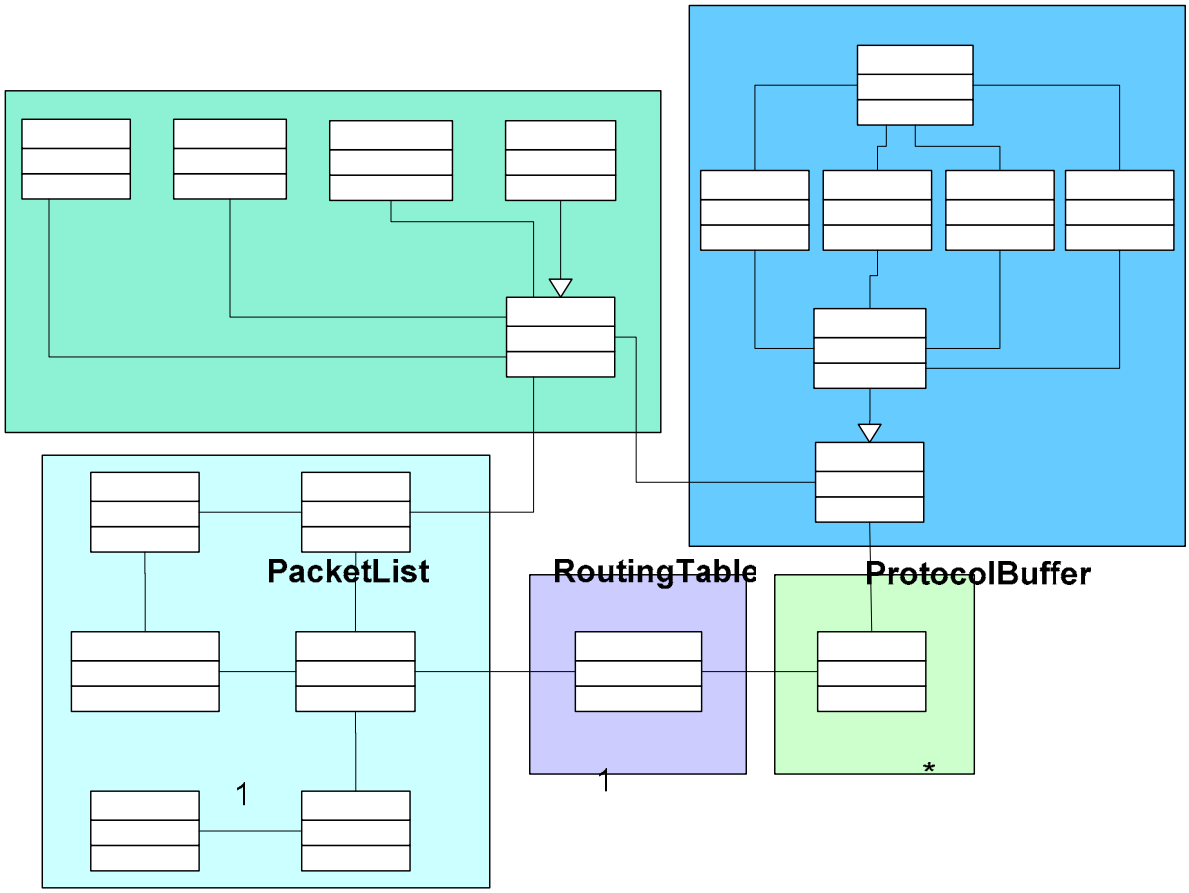


Figure 4-1 AHS UML Design

The AHS program consists of different components:

- Neighbour Searching
- Media Access Control
- Protocol
- Network
- Graphical part / PDA Container

The neighbour searching component uses the sector technique described in the 3-5-5 section. Because a computer can not easily determine which coordinates are within a certain range a neighbour searching algorithm will be implemented. If no neighbour searching routing algorithm is implemented all coordinates must be checked if they are within a specific range of a coordinate. This takes a lot of processor time and it unnecessary.

The Media Access Control (MAC) is responsible for moving data packets to and from one Network Interface Card (NIC) to another across a shared channel. In the simulation the standard IEEE (Institute of Electrical and Electronics Engineers) 802.11b will be implemented. IEEE 802.11b is also referred as 802.11 High Rate or Wi-Fi.

The protocol decides how a message must reach the destination. There are not yet standards, so a protocol has to be designed. A user only has to decide where the message must go to and the protocol tries to deliver the message.

The network class attempts to connect to the AHV program. The incoming traffic is given to the PDAContainer that processes the message and takes action.

The graphical part and the PDA container provide a user interface and manage the other components. Information of a PDA can be requested and displayed. It also allows the user to send a message to other PDA's. The communication traffic then will be visible in the AHV program.

When the AHS program is started it immediately start to attempt to connect to the server. Then the networksocket will listen to incoming network traffic. The PDAContainer then will process incoming data. In case a property of the PDA must be adjusted or added a new instance will be created in the PDA class. The PDA class start a Protocol process and the protocol class starts the MAC process. The MAC class will have a static reference to the Neighbour Search component.

The graphical window is started when the Networksocket is started. The user sees real-time updates of PDA's. Once a PDA is selected more information can be viewed. Furthermore the user is permitted to modify configurations and send message from a PDA to another.

After the message is send the protocol will send this to the MAC. The MAC receives the destination id, source id, the previous/destination id, the data, the more and mf fields. The more field is enabled if more packets for the next node are waiting and the MF field is enabled when more fragments are coming that belong to the same set. The previous/destination field has the value 99999, if the message is broadcasted or else it can be set to the next node's id. The MAC will then control the delay of the packets. Incoming packets then will be given to the Protocol class. Ultimately a complete packet can be viewed in the graphical window.

4.2.2 AHV

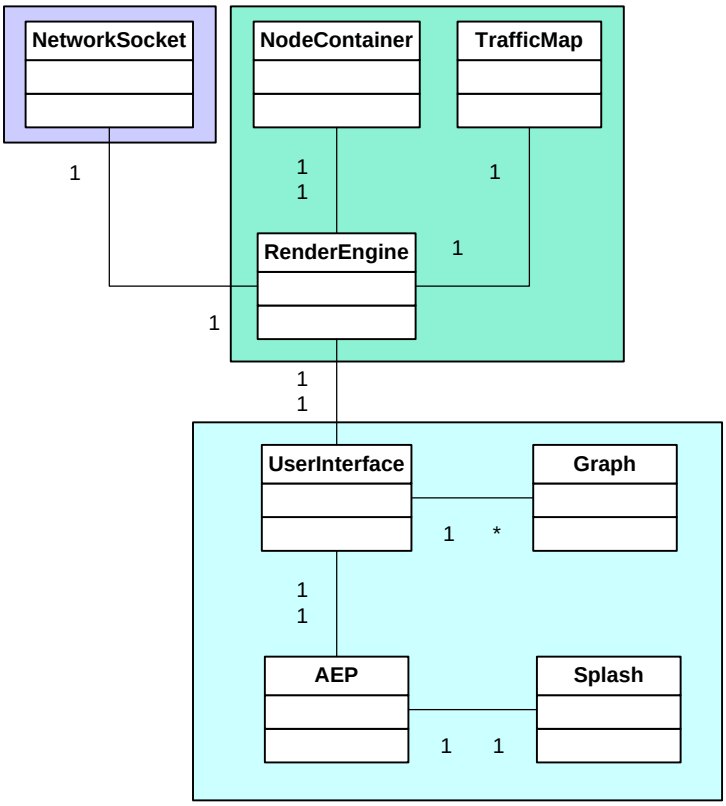


Figure 4-2 AHV UML Design

The AEP class starts the AHV program by creating a new instance of the User Interface class. The User Interface draws a new window a will take care of the input of the user. When the simulation is started also a graph will be drawn on the window. The RenderEngine class is the most important class in the AHV, because almost all actions take place in here for example; what actions must be taken if it receives network data. Of course the Network class has also functionality, but doesn't decide the actions. It will only listen for incoming data and allow data to be send over the network, but received data will be given to the Render Engine class, which decide what to do. The Nodecontainer contains all the locations of the nodes and potential information like the id. The TrafficMap has all the functionality that will draw the map and the moving traffic; all graphical matter takes place in here.

When the program has been started the RenderEngine will not do anything yet. Once a Dataset has been loaded it automatically loads the map and calculates how many pixels equal 550 meters. This range is the maximum range of the PDA. This range is used to transform pixel locations into real location. Also when the traffic is moving, circles can be drawn to indicate the ranges. However if the maximum range of 550 meter is more than a fifth of the map size it won't be drawn.

The TrafficMap now holds the image of the traffic map without the traffic drawn on it. The simulation must first be started to display the moving traffic on the map. If the user starts the simulation, the network socket will listen to incoming network traffic and the AHS program starts and will be minimized. The AHS program tries to connect with the AHV program. If it fails it tries again after a few seconds. The AHV program continues when a connection has been set up with the client.

At this point the Render Engine reads a line of the DataSet file when the started timer expires. The information of the line is being extracted and a part of the information is transformed into coordinates see section 5.4.2. The coordinates will be placed into the NodeContainer class and the real coordinates in meters are being send to the client. When the end of the tick/ timeline is reached, RenderEngine will give all coordinates in the NodeContainer to the TrafficMap. Which draws all the traffic. The reading of the file will carry on until the end of the file has been reached.

When the program is started, it also will listen to incoming network traffic. This network traffic can influence existing data in the classes. But it will not interfere with the simulation process itself.

A difficulty in the design is when the program is being stopped, all processes must be also stopped. If not all processes are being ended unwanted background processes could still be running or memory is not cleared nicely.

4.2.3 Socket Communication

To send a message over socket first a connection must be made. The default port is 9001 and the default host address will be 127.0.0.1. The program will automatically try to establish a connection. When it fails to establish a connection it will try again after a few seconds until it will succeed. The AHV will act as the server and the AHS will act as the client.

The messages that are being send over the socket all have the same structure, see figure 4-3. First the command must be given such as add a node. After the command a double dot character follows then several variables can be given separated by a '|' character.

```
[COMMAND]:[VAR1]|[VAR2]|…
```

Figure 4-3 Socket Command Structure

In the table 4-1 all available commands are given. Be aware that the commands are case sensitive.

Table 4-1 Socket Commands

Command	Variables	Description
ADD	ID|X|Y	Adds a node
DEL	ID	Removes a node
MOV	ID|X|Y	Moves a node
CREATE	WIDTH|HEIGHT	Create an area
SEND	FROM_ID|TO_ID|MESS	Send a message to someone
RECEIVED	MESS	Returns the incoming message
COMM	SOURCE_ID|DEST_ID	Communication between nodes

4.2.4 Vehicle Routing Information

There are two ways of extracting vehicle routing information of the City program. By sending real-time data over the network or storing all information in a file and read the coordinates when the simulation has ended. However both approaches have shortcomings. Sending real-time information can increase the network communication and cost a lost of processor time, but sudden changes in routing can real-time be noticed in the AHV program. By storing all information in a file real-time vehicle movements is almost impossible, but when the simulation is completed the file can be optimized and can save processor time. Because coordinate transformation can be computed before the simulation and vehicles that are drawn on top of each other or drawn twice can be eliminated. The second solution, which saves data to a file, has been chosen, because it can optimize the data, which saves a lot of processor time, which probably will be essential in the AHV and AHS program.

4.3 Programming Style

A programming style has been chosen which makes the code easier to read. A few rules have been created. The first rule has to do with the variable names. If a variable is readable in the entire class it has got an '`m_`', a member class variable, before the variable name, for example variable address is converted into `m_address`.

Especially large code is difficult to read and will be complicated when it is not structured well. In the programming editor Visual Studio .NET regions can be used. The `#region` command can minimize a block of code. Somewhere in the code the line '`#region region_name`' must be added and at the end of the block code `#endregion` must be added. Now a minus (⊟) will appear. If clicked on it the ⊟ will change into ⊞ and all the code between the block will be invisible only the region name is now displayed. The view the code only a double click with the mouse makes the code visible again. It is also possible to create a region within a region.

Sometimes regions are not enough to apply a structure in the code, which is why small comments have been added. Code can now easily be found. An example can be found in table 4-1, this shows where the constructor is located in the code.

```
//---------------------------------------------------------------------
// Constructor
//---------------------------------------------------------------------
```

Figure 4-1 Style in Code

C# provides a mechanism for developers to document their code. This looks a bit like Java documentation, but it uses xml instead of html. In table 4-3 a sample can be found how a method can be documented. Documentation can be generated by executing this command '`csc filename.cs /doc:documentfilename.xml`' (see [XDD] for more information).

```
/// <summary>
/// Calculates the sum.
/// </summary>
```

```
/// <param name="nr1">First number</param>
/// <param name="nr2">Second number</param>
/// <returns>the sum of the two numbers</returns>
private int Sum(int nr1, int nr2) {
       return nr1 + nr2;
}
```

Figure 4-2 Generate documentation

Finally, source code files can be placed in solution, directories or namespaces. This organizes the structure of the files. Directories can be used to manage a structure in the project. It doesn't really matter if a file is placed in a different directory. A solution can contain more projects sometimes this can be handy when working with more than one project. Classes can be placed into namespaces. If a class name is used twice, but exists in different namespaces it still can be compiled. The usage of namespaces is very useful to create components.

Chapter 5 : Implementation

5.1 Justification

All the researches, documentation and implementation have been carried out by me. Except the MAC 802.11b file is replaced by Joost L. Boehlé's version and the Thread Base class of Joost L. Boehlé is used.

5.2 Techniques

Software can be implemented in different ways even if it uses the same designs. For example searching for neighbours, as explained in section 3.3, can be implemented using different techniques. The simplest technique is usually the worst one. But efficiently techniques are often more complex to implement. Though it increases the performance of the program. In the next sub sections some techniques will be discussed that have been implemented.

5.3 AHS

The next sub sections will discuss which methods and configuration the Ad Hoc Simulation program contains.

5.3.1 ARA Implementation

As mentioned in the section 3-3, ARA must be modified or extended to be implemented in the program. For example ARA does not define the structure of the message and the message can be forwarded forever in a large network. Another disadvantage of ARA is that, if the data packet has arrived at the destination it will not reply with an acknowledgment, that the data has arrived. The source will never know if the data has reached the destination. ARA has been extended with a small packet named ACK. The ACK packet will be send every time that the destination receives a data packet. The source that has transmitted the packet to the destination waits until it receives the ACK packet or if it expires it will retry to resend the packet a certain number of times.

Let's examine how the structure of an Acknowledgment packet (ACK) looks like. The first eight bits TTL, stands for Time To Live, this field is used to stop the message from travelling all the way through the network. Let me clarify the method, each time it bypasses a node the TTL field will be decremented by one. The message can travel around the network until the TTL becomes zero, then the message will be not be discards. The maximum number of nodes that can be visited is 256 (2^8).

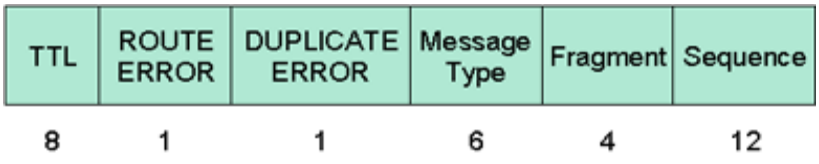


Figure 5-1 ACK packet (in bits)

The second field indicates if the message successful has arrived at the next node. If the value of the route error field is one then the next node has not successful received the message. The duplicate error

field is used when the next node already has received this packet and will reply the ACK back to the previous node with the duplicate error flag 1. The previous node now knows that the message has not arrived successful.

There are four different message types, which can be distinguished by the different values see table 5-1.

Table 5-1 Packet types

Type	Value
BANT	0
ACK	1
FANT	16
Data File	32
Data Text	48

The last two fields of the ACK message are used to recognize if the message has already been received. The maximum sequence number is 4096 (0-4095) and the maximum fragment number is 16 (0-15).

The BANT, FANT and Data messages have got the identical message structure as in figure 5-2. The only difference with the structure of the ACK message is that the Data field has been added.

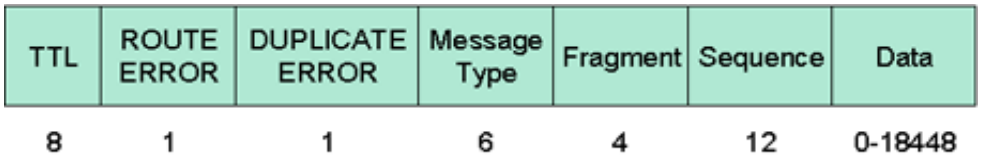


Figure 5-2 BANT, FANT and Data packet structure (in bits)

However the first Data file packet divers from this structure. One field is added to the data field. This extra field stands for the filename of the file that is being send. It has the length of 256 bytes this equals the maximum number of characters in windows for a filename.

5.3.2 Default Configuration

The protocol and the MAC that has been implemented uses some default variable values; the broadcast address in the MAC is 255.255.255.255.

The packet will be send three times after the third time the message will be reported as failed. Also a timer will be started when the message is being send. It waits for a maximum time of 2 seconds. If the timer expires the message will be dismissed. But if a reaction has been received the timer will be adjusted according to the result of the formula $waittime = minwaittime + \Delta timetoreplyinsec$. Because the timer is adjusted, the retry procedure can be performed earlier now.

5.3.3 Building messages

A short text or file message will only take one packet, but packets larger than 2312 bytes must be send in more than one packet. Building up a text message is not that difficult, each time it combines the data in the packet with the packet in memory. However a file transfer is more difficult.

A file larger than 2306 bytes will be spitted into fragments of 2306 bytes. Each fragment will be send independently. When it arrives at the destination it is saved on the hard disk. Finally all fragments are combined to one file. File names contain the sequence number and fragment number and an increment number. The *increment number* can be found by looking first if the fragment and sequence filename already exists. If files are found, the highest increment number is taken and added by one or else the increment number is set to zero. This approach has got the advantage that when a fragment is delayed the file still can be correctly made if it is within the time of next sixteen fragments.

Two other possibilities to create a file are by adding each time the bytes to the file or creating the total sized file and later on fill it with bytes. Adding each time bytes to a file has a major disadvantage; every fragment must be acknowledged before the next fragment can be send. By filling a file with bytes a location must be given where the bytes must be placed. It offers more reliability than the implemented approach. However it will takes more bytes that must be send. Especially when vehicles transfer large data to each other and are moving it can just make the difference between failing and succeeding.

5.3.4 Clean up

Every PDA has some memory and has been given some hard disk space. If the maximum memory or storage usage has been reached other data will be overwritten. Because the program needs much processor power, it cannot check every few seconds for old data that must be removed from memory. But it must check for old data or else the memory usage will increase infinitely. So a simple solution has been created to solve this problem. Every time it searches through the elements it will check if the packet is recent. The current time will be measured up to the time the element was created. Sometimes an element must stay longer in memory this can be realized by updating the creation time of the element. The code in figure 5-1 shows the implementation.

```
DateTime now = DateTime.Now;
System.TimeSpan TS = new System.TimeSpan(now.Ticks-lastupdated.Ticks);
int sec = Convert.ToInt16(TS.TotalSeconds);
if (sec > maxtime)
      return false;
else
      return true;
```

Figure 5-1 Cleaning memory

5.3.5 Byte Wise

All messages are translated into bits. But the C# programming language doesn't allow single bits types; instead it uses bytes (8 bit). Text characters are not difficult to convert because every character is converted to an ASCII (spoken as ask-ee) number that equals one byte. ASCII, American Standard Code for Information Interchange, is a code for representing English characters as numbers. A major problem is when a 12 bits field and 4 bits field must be created and extracted see figure 5-3. This is the situation when the sequence and fragment field must be created. The remaining 4 bytes of the 4 bits field belong to the 12 bits field. The 4 bytes must be separated and combined with the other 8 bits.

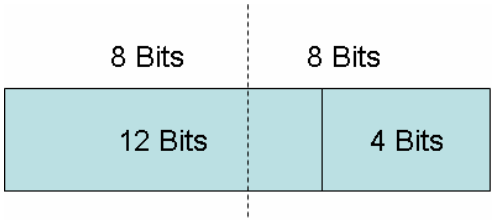


Figure 5-3 Sequence (12 Bits) and Fragment (4 Bits) field

See table 5-3 for the code to generate the two bytes and to separate the numbers. This procedure is executed when a Fragment field and Sequence field is composed.

```
// Set the sequence and fragment number
Byte[] tmp        = IntToByte(sequence);
DataFile[3]       = tmp[0];
DataFile[2]       = (byte)(fragment<<4);
DataFile[2]       += tmp[1];


// Get the sequence and fragment number
byte[] bytes = new byte[2];
bytes[0] = inbytes[2];
bytes[1] = inbytes[3];

fragment = (int)(bytes[0] >> 4);
bytes[0] -= (byte)(fragment << 4);

byte t = bytes[0];
bytes[0] = bytes[1];
bytes[1] = t;

sequence = ByteToInt(bytes);
```

Figure 5-2 Set/Get sequence and fragment number

5.4 AHV

The main issue of Ad Hoc Visualization program is how the coordinates can be extracted in an efficient way. The next subsections will give more information of how the coordinates are being processed.

5.4.1 Coordinate Transformation

Coordinates can easily be extracted from the city program by using the screen pixel coordinates, however there are two major problems. The first problem is that the map is being stretched. The map height and width will be adjusted so that it will perfectly fit in the graphical window. However the map is not displayed like in the reality. Some roads look relatively longer. It must be taken into account that distance calculation cannot be done be simply calculating the distance between two screen coordinates. The coordinates must first be converted to real coordinates.

The second problem is that the city program draws vehicles next to the road instead of drawing the vehicles on top of the road. Distance calculation is also a problem now because it won't be precise. Suppose two vehicles are driving on the same road in the opposite direction and are just in each other's communication range. It is possible that they cannot communicate, because of the distance they are drawn from the road.

As mentioned earlier the map in the city program is stretched (try some resizing of the window), however this will not be the case the AHV program. The height and width of the map will be equally increased until the width or height reaches the size of the graphical drawing component, see figure 5-4.

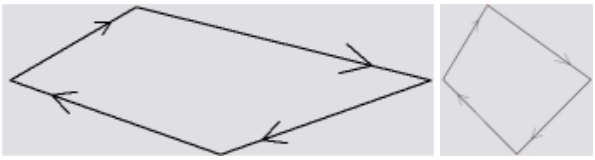


Figure 5-4 City Program Map (left) and AHV Map (right)

5.4.2 Dataset Files

The city program has been adjusted so that information of the vehicle movements are stored in a file. The information that is being stored looks like figure 5-5. This coordinate information file, dataset file, has the extension `.dsf`. Every tick a line like '--' is being stored in the file between these lines the coordinate information can be found. The dataset file `.dsf` is automatically generated when the City Program has started a simulation. The files can be found in `c:\`.

Other generated dataset files by Joost L. Boehlé with extensions such as `.raw` and `.ORG.ds` contain different structure formats. But can also be used in the AHV program. I recommend for coordinate precision to use the `.dsf` files.

```
id|x|y|relposlink|curlink|edgecolor|innercolor|vehiclesize
```

Figure 5-5 Structure DSF Files

The `.map` files contain all roads and intersections coordinates. There are two ways to store this information in the memory, by not yet converted coordinates to screen or the converted coordinates. If they are drawn on the screen it will result in the same graphics. However the not yet converted coordinates require an extra calculation before they can be drawn on the screen.

Ultimately meters per pixel must be calculated. This variable is used to measure how many pixels the maximum range will be and the distance in meters between two points. The distance between the start coordinate and the end coordinate will be calculated and then be divided by the length in meters.

5.4.3 Graph

The graph has been implemented to prove that the routing algorithm actually works. The graph is no component from the internet, but has been build by me. This because the graph is constantly updating itself and most found on the internet don't take that into account. The implemented version sets both rulers(horizontal/vertical lines) to the maximum value ten. If a value comes above ten, let say fifteen, the rulers will be redrawn by setting the maximum value of a ruler to twenty five (ten will added, but this value can be changed). This way not every time when the graph is being updated that rulers will redrawn only the lines.

The graph only needs a value and assumes that horizontal value is the time. Two extra functions have been added; Bitmap Save and Statistic Save. The Bitmap Save will save the image to a file. The Statistic Save will save all the information of the line into a file. The file will have an extension '`.csv`' .This allows programs such as Microsoft Excel, Open Office and StarOffice to examine the results. The code and explanation can be found in the appendix ….

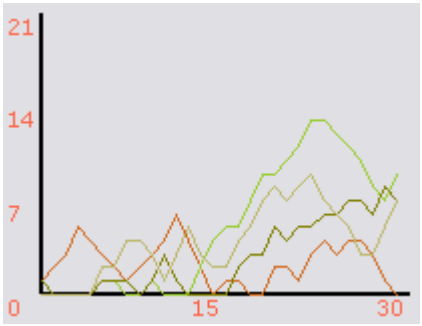


Figure 5-6 Graph

5.5 Thread Base

Note Joost L. Boehlé has written this class. It uses the functionality of a Thread class, but there are some differences. The override launch method is used to implement what the process must do if it is started. The EnterWait method enables to block the thread process until the method ExitWait is called. This class is used in the MAC class and Protocol class.

5.6 Specification

The minimal requirements of the AHV program are; 500 MHz, 256 MB RAM and Video Card. The AHS on the other hand requires a more powerful system, 800 MHz, 320 MB Ram (512 Recommended). Both programs must be connected to a network, so both programs can communicate with each other. It is important not to run the program as a process when currently working with other intensive programs.

5.7 Installation

There is a setup file that starts the installation. Follow the instructions and all necessary files will be installed. An alternative is to download the .NET Framework 1.1 [DNET] then start the executable file of the AHV program.

5.8 User Interface

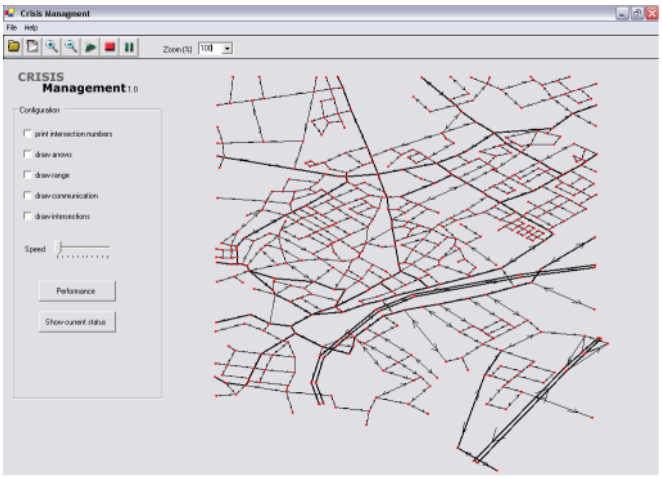


Figure 5-7 AHV 1.0 User interface Screenshot

All graphical files such as the buttons and the splash screens have been made by in Adobe PhotoShop. The files can be modified or adjusted. Pay attention, when changing or adding a self-made image, the Bitmap format is the best option. If other image formats are chosen quality of the image may decrease.

The user interface design allows the user to directly influence the program settings:

- Arrows can be turned on or off.
- The intersection id's can be displayed.
- The colour of the road can be set.
- Background colour of the map can be set.

All generated technical documentation as described in section 4.3 can be found in Menu > Help in both the AHS and AHV program.

Figure 5-8 Screenshot AHV 1.0

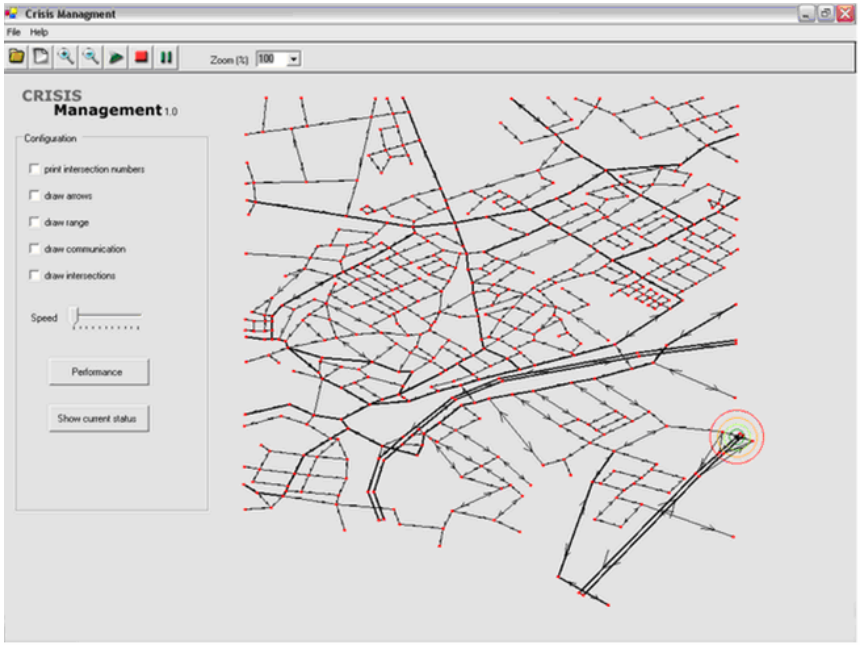
When a PDA is selected, information can be requested of the MAC. Information of the protocol will be automatically displayed when a PDA is selected. It is also possible to save the statistics in a file.

Chapter 6 : Test

6.1 Features Program

- Reading .raw, .ORG.ds and .dsf dataset files.
- Graphical representation of the vehicles, the map and the communication.
- IEEE 802.11B standard is implemented.
- ARA Protocol is implemented.
- Adjustable graphics (arrows, intersections etc.).
- Configuration, settings can be modified.
- Socket communication.
- Saving statistics.

6.2 Running simulation



To start a simulation first a map must be chosen. The map will be drawn onto the screen. The nodes will be drawn on the map if the user pushes the start button. When nodes are in range of each other a line between the nodes will appear. If they are communicating a dotted line will be drawn instead of the line.

The user can change the graphics even when the simulation is started. Some graphical options that can be changed are zoom, drawing arrows on the road, displaying the communication range of a node etc. Also statistic can be saved into a file and the graphics can be saved in many image file formats. The AHV can also save statistics, but is mostly used to organize and exchange information.

6.3 Results

What will happen if ARA is up and running? This is the main question of this section. In the next example an area has been taken with a width and height of 1500. There are 30 PDA's and all of them have a range of 500. The density is high. The simulation sends a short text message containing "ping" every second.

As we can see in the left figure 6-1 the send data traffic in proportion to the received data traffic is less. The horizontal line indicates the time in seconds and the vertical line indicates the number of messages. After a while you can notices almost every line is stabilising. The high density of PDA's causes the heavy data traffic. The results in figure 6-1 were as excepted. The right figure 6-1 shows how ARA will react if no PDA's are within each other's range.

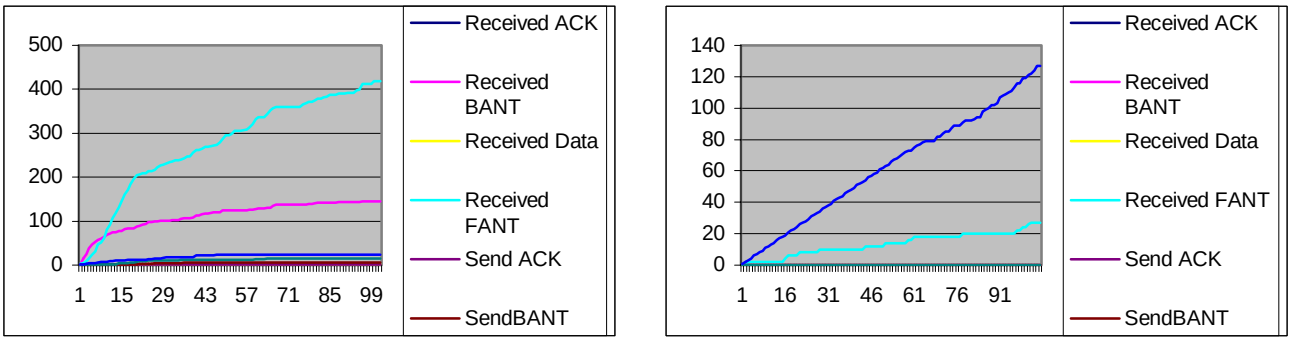


Figure 6-1 ARA Result (right side high density, the left has a very low density)

Chapter 7: Evaluation

7.1 Evaluation

In this chapter recommendations, future work and conclusion will be discussed.

7.2 Problem definition

As described in section 1.2 there are different problems:

1. How to create a route? (section 3.3.4)
2. What will be the structure of the message? (section 5.3.1)
3. How will the message be processed? (section 5.3.1)
4. How is information being distributed over the network? (section 5.3.1 & section 3.3.4)

Most answers to the above questions be found in section 5.3.1. The routing protocol ARA solves a lot of the problems. However the another layer must decided the frequency and the content of the information. The Agent Layer can provide a solution by letting agents travel over the network to search information.

7.3 Recommendations

Although the program is up and running and can be used there are some improvements to create more realism, security and quality of service. Also it is possible to use components of the program, but this is up to you. If you do so please inform the administrator of this project.

7.3.1 Security

The simulation program doesn't use security techniques. However in practice, security will be an important issue. There are not many methods that offer security in a MANET. Especially because attacks on MANET's defer from Internet attacks.

Ariadne prevents attackers or compromised nodes from tampering with uncompromised routes consisting of uncompromised nodes, and also prevents a large number of types of Denial-of-Service attacks [ARID].

A simple solution to offer is the Hoffman method. An implementation can be found in the appendix. First keys must be exchanged before the crypt message can be send. The performance depends on the number of characters in a message, for example if there are 500 characters it transforms every character. A standard library is used but can of course be modified or created dynamically.

7.3.2 Realism

To create a more realistic simulation some features must be implemented. The PDA's already have memory and a hard disk, but doesn't use a battery [BAT]. This causes, that a PDA that is travelling always has enough energy.

The implemented MAC that uses the standard IEEE 802.11b misses the functionality of CRC, Cyclic Redundancy Check. This field is used to determine if a packet has arrived successful.

7.3.3 Quality of Service

Providing suitable quality of service (QOS) support for the delivery of real-time audio, video and data in mobile ad hoc networks presents a number of significant technical challenges. A key component of INSIGNIA is an in-band signalling system. The signalling system is designed to be lightweight and highly responsive to changes in network topology, node connectivity, and end-to-end quality of service conditions [IIN].

7.4 Future Work

There are different possibilities of using this program. The next sub sections will give some suggestions how the program can be applied in the future. But other plans can also be developed.

7.4.1 Agent Platform

An agent platform allows agents to travel trough the network. An agent is a program that performs some information gathering or processing task in the background. Typically, an agent is given a very small and well-defined task.

I recommend implementing this by connecting the protocol class to the agent platform. Agents in the platform can process messages that are processed by the MAC and the protocol. However there are many other possibilities of implementing an agent platform and combining this with the AHV/AHS program.

There are a number of possibilities of adding an agent platform to the simulation; by a socket connection, by extending the program or running the program as a different process on the same computer and exchanging information by memory. If the extension option is chosen the language must be a .NET language or Java (known as J# in .NET).

7.4.2 Module

The AHS program can be used as a module. Suppose, a program wants to use the functionality of the AHS program, it will have to act like a server. The default port is 9001 and host location is 127.0.0.1, but can be modified. The program that acts as the server can start the AHS program, which automatically tries to connect. When a connection is established the data can now be send. The commands that can be send, can be found in section 4.2.3. Two methods can be used to accomplish this, by starting up the program before the program is necessary or by start it up when necessary. This is up to the other program.

The previous paragraph discussed how the functionality of the program could be used, by using the socket. But it is also possible to use some modules of the programs. The most recent documentation of the functionality can be requested in the AHS and AHV program. Some examples of modules that can be extracted are; NetworkSocket, NeighbourSearching, UserInterface AHV, engine AHV, MAC and ARA (RoutingTable, RoutingBuffer, PacketList). This approach saves a lot of networkcommunication, but is more complex.

Functionality can also be placed on PDA's [MCE]. The MAC and neighbour searching algorithms are not necessary anymore. But the (ARA) Protocol is very important for the route searching. Some adjustments have to be made like the network communication, memory and hard disk usage and user interfaces must be modified.

7.5 Conclusion

There are a lot of idea's, methods and proposals for MANET's, but only a few of them provide efficient methods. And these must then be adjusted or combined with other methods. This process makes it very difficult because maybe there is too much information available except a complete solution.

The routing protocol ARA provided correct to work in this project. The results are promising, but the statistics must be compared with other paper results. Of course these results cannot always be trusted. Another thing to be aware of is that the program doesn't simulate interference of the signal. So a maximum range can always be reached. However in practice this probably will be less.

There are lot of possibilities of applying MANET's. They not only provide advantages, but also disadvantages. But for spreading information combined with the ARA routing protocol it offers a lot of opportunities to be applied in reality.

Appendix A : IEEE802.11b Frames

IEEE 802b Frames

All IEEE 802.11 frames are composed by 6 fields. The data field contains the MAC Data. The frame control field in the MAC Data contains 11 fields.

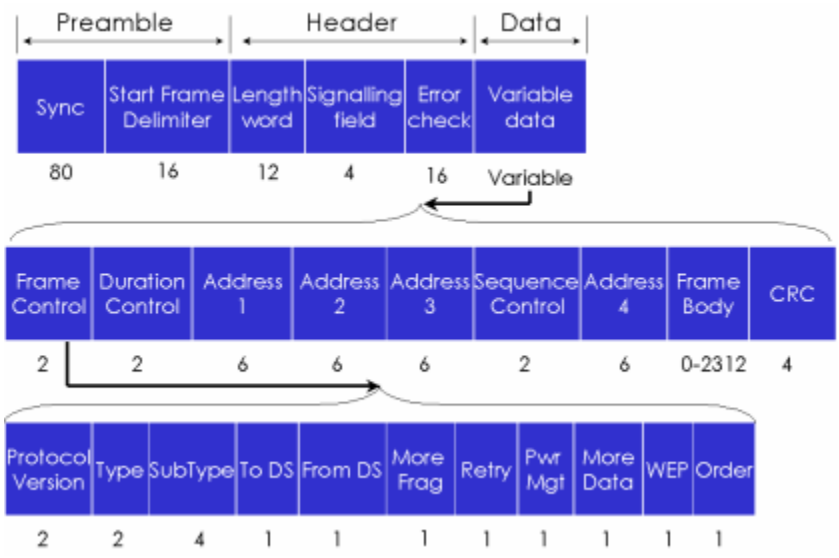


Figure 7-1 IEEE 802.11b Frames

Frame

A frame is composed by a few components:

- Preamble:
  - Sync; is used to detect a potentially receivable signal, select an antenna if diversity is supported and acquire symbol timing.
  - Start Frame Delimiter (SFD); is used to define the frame timing
- Header:
  - Length word; is used for correctly detecting the end of the packet.
  - Signalling field; is used to indicate the data rates from 1 Mbit/s to 4,5 Mbit/s in 0,5 Mbit/s increments.
  - Error check; a 16 bit Cyclic Redundancy Check (CRC) error detection field [CRC] [UCRC]
- Data; contains the MAC Frame Format (see … MAC Data)

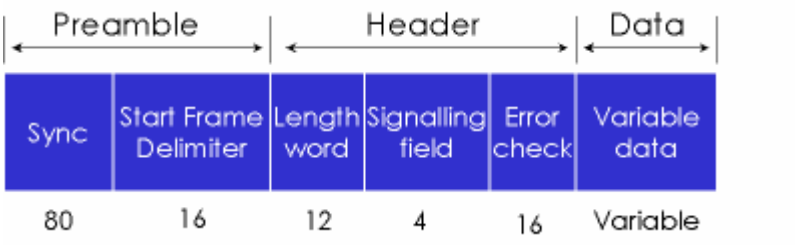


Figure 7-2 Frame

MAC DATA

The MAC Frame Format contains:

- Frame Control; see Frame Control
- Duration Control; contains the data on the duration value for each field and for control frames it carries the associated identity of the transmitting station.
- Address 1; the recipient address
- Address 2; the transmitter address, how physically is transmitting the packet.
- Address 3; in most cases it is the remaining, missing address, on a frame with FromDS set to 1, then the adress3 is the original Source Adress, if the frame has the ToDS set then the Address 3 is the destination Address.
- Sequence Control; is used to represent the order of different fragments belonging to the same Frame and to recognize packet duplications. It contains two subfields “Fragment Number” and “Sequence Number”, which define the number of the fragment in the frame.
    - Sequence Number; a 12-bit field, from 0 to 4096 and incrementing by 1
    - Fragment Number; a 4-bit field, the first fragment will be zero and the following fragments are incrementing by 1
- Address 4; is used in special cases where a Wireless Distribution System is used, and the frame is being transmitted from one Access Point to another. Both the ToDS and FromDS bits are set, so both the original Destination and the original Source
- Addresses are missing.
- Frame Body; -
- CRC; a 32-bit CRC, which ensures there are no errors in the frame.

Figure 7-3 MAC Data

Frame Control

The frame control field contains:

- Protocol version; the protocol version field is 2 bits in length and will be used in to recognize possible future versions. The value of the current version is initialized as 0.
- To DS; is set to 1 when the frame is addressed to an AP for forwarding to the distributed system, else it is set to 0.
- From DS; is set to 1 when more fragmentations belong to the current fragment.
- More Fragments; is set to 1 when more fragments belonging to the current fragment.
- Retry; indicates if the fragment is a retransmission of a previously transmitted fragment. This will be used to recognize duplicate transmissions when an acknowledgment packet is lost.
- Power Management; indicates in which state the node will be in after the transmission of the packet.
- More Data; used to indicated that there are more frames buffered for this node.
- WEP; indicates that the frame body is encrypted according to the WEP algorithm. This will not be implemented (default will be 0).
- Order; is set to 1 if the frames must be strictly ordered. This will not be implemented (default will be 0).

Figure 7-4 Frame Control

Type / Subtype

There are three types of frames; control frames, management frames and data frames. Data Frames are used for data transmission. Control Frames are used to control access to the other nodes. Management frames are used to exchange management information, but are not forwarded to upper layers. Some examples are Beacon, Probe and Power Management.

Table 7-1 Frame types/subtypes

Type	Type Description	Subtype	Subtype Description
01	Control	1011	RTS
01	Control	1100	CTS
01	Control	1101	ACK
10	Data	0000	Data
00	Management	0100	Probe Request
00	Management	0101	Probe Response

RTS Frame

Figure 7-5 RTS Frame

The duration is the time, in microseconds, required to transmit the next Data or Management frame, plus CTS frame, ACK frame plus three SIFS intervals.

CTS Frame

Figure 7-6 CTS Frame

The Receiver Address is copied of the CTS frame from the Transmitter Address field of the previous RTS frame.

Duration = Duration field RTS – time in microseconds required to transmit the CTS frame and its SIFS interval

ACK Frame

Figure 7-7 ACK Frame

The receiver address is the address from the previous frame. If the More Fragment bit was set to zero in the Frame Control field the duration is set to zero, else the duration is duration of the field of the previous minus the time to transmit the ACK frame and its SIFS interval.

Appendix B : Performance Neighbour Searching

Area	(0,0)-(300,300)
Depth	5
Nr PDA	5000
Moving	5000
Removing	10
Searching	5000
RangePDA	10

Technique	Sector	QuadTree xor	QuadTree walk	None
Insert	0,01	0,01	0,02	0,01
Move	0,02	0,01	0,05	0
Deleting	5,70E-05	0	0	0
Search	0,59	3,96	22,12	24,85

Technique	None	QuadTree walk	Sector	QuadTree xor
Insert	0,01	0,02	0,01	0,01
Move	0	0,05	0,02	0,01
Deleting	0	0	5,70E-05	0
Search	24,85	22,12	0,59	3,96

Appendix C : Sector

```
using System;
using System.Collections;
using System.Drawing;

namespace QuadTree_csharp {
	/// <summary>
	/// 2D storage for points includes a neighbour searching algorithm
	/// </summary>
	public class Sector {
		private TreeNode[,] m_treeNodesArray;
		private ArrayList m_treeNodes = new ArrayList();

		private float m_opt_size=0;

		private Point m_XYTop;
		private Rectangle m_area;

		private int sizearrayx;
		private int sizearrayy;

		public Sector(Rectangle area, int static_range) {
			//width area's = range*2.5

			m_opt_size = (float) (static_range*2.5);
			sizearrayy = (int) (area.Height / (m_opt_size));
			sizearrayx = (int) (area.Width  / (m_opt_size));

			m_area = area;
			m_XYTop = new Point(area.X,area.Y); // left top
			m_treeNodesArray = new TreeNode[sizearrayx,sizearrayy];
			Point pnt = m_XYTop;
			for (int x=0;x<sizearrayx;x++) {
				for (int y=0;y<sizearrayy;y++) {
					Rectangle tmparea = new
						Rectangle((int)(x*m_opt_size),(int)(y*m_
						opt_size),(int)m_opt_size,(int)m_opt_siz
						e);
					m_treeNodesArray[x,y] = new
						TreeNode(tmparea,null,null,null,null);
					pnt = new Point((int)(y*m_opt_size),pnt.Y);
				}
				pnt = new Point(pnt.X,(int)(pnt.Y+m_opt_size));
			}
		}

		/// <summary>
		/// Adds a pda.
		/// </summary>
		/// <param name="pda">The pda that must be added</param>
		public void Add(PDA pda) {
			TreeNode tmp = GetNode(pda);
			tmp.m_elements.Add(pda);
			pda.TreeNode = tmp;
```

```
        }

        /// <summary>
        /// Removes a pda.
        /// </summary>
        /// <param name="pda">The pda that must be removed</param>
        public void Remove(PDA pda) {
              TreeNode tmp = GetNode(pda);
              tmp.m_elements.Remove(pda);
        }

        /// <summary>
        /// Changes the location of the pda.
        /// </summary>
        /// <param name="pda">The pda that must be moved</param>
        /// <param name="newlocation">The new location</param>
        public void Move(PDA pda, Point newlocation) {
              Remove(pda);
              pda.Location = newlocation;
              Add(pda);
        }

        /// <summary>
        /// Returns the node where the pda is located in.
        /// </summary>
        /// <param name="pda">The pda that must be found</param>
        /// <returns>Node where the pda is located in</returns>
        public TreeNode GetNode(PDA pda) {
              int x = (int)((pda.Location.X-m_XYTop.X)/m_opt_size);
              int y = (int)((pda.Location.Y-m_XYTop.Y)/m_opt_size);

              return m_treeNodesArray[x,y];
        }

        /// <summary>
        /// Searches all neighbours of the pda.
        /// </summary>
        /// <param name="pda">The pda to search</param>
        /// <returns>The location</returns>
        public ArrayList SearchNeighbours(PDA pda) {
              int x = (int)((pda.Location.X-m_XYTop.X)/m_opt_size);
              int y = (int)((pda.Location.Y-m_XYTop.Y)/m_opt_size);
              ArrayList nlist = new ArrayList();

              nlist.Add(m_treeNodesArray[x,y]);

              bool xl = false, xr = false;
              if (x-1 >= 0 && pda.Location.X - pda.TransmissionRange <
                    pda.TreeNode.m_area.X) {
                    // located on the left side of the rectangle
                    nlist.Add(m_treeNodesArray[x-1,y]);
                    xl = true;
              } else if (x+1<sizearrayx && pda.Location.X +
                    pda.TransmissionRange > pda.TreeNode.m_area.Right) {
                    // located on the right side of the rectangle
                    nlist.Add(m_treeNodesArray[x+1,y]);
                    xr = true;
              }
```

```
                if (y+1 < sizearrayy && pda.Location.Y + 10 >
                      pda.TreeNode.m_area.Bottom) {
                      // located on the lower side of the rectangle
                      nlist.Add(m_treeNodesArray[x,y+1]);
                      if (xl)
                            nlist.Add(m_treeNodesArray[x-1,y+1]);
                      if (xr)
                            nlist.Add(m_treeNodesArray[x+1,y+1]);
                } else if (y-1 >=0 && pda.Location.Y - 10 <
                      pda.TreeNode.m_area.Y) {
                      // located on the upper side of the rectangle
                      nlist.Add(m_treeNodesArray[x,y-1]);
                      if (xl)
                            nlist.Add(m_treeNodesArray[x-1,y-1]);
                      if (xr)
                            nlist.Add(m_treeNodesArray[x+1,y-1]);
                }

                ArrayList neighbours = new ArrayList();

                for (int i=0;i<nlist.Count;i++) {
                      TreeNode tmp = (TreeNode)nlist[i];
                      for (int u=0;u<tmp.m_elements.Count;u++) {
                            PDA pdatmp = ((PDA)(tmp.m_elements[u]));
                            float distance = Distance( pdatmp.Location,
                                                      pda.Location);

                            if (distance <= pda.TransmissionRange && pda
                                  != pdatmp)
                                  neighbours.Add(new
                                              Neighbour(pdatmp,distance));
                      }
                }

                return neighbours;
          }

          /// <summary>
          /// Calculates the distance between two points.
          /// </summary>
          /// <param name="point1">The first point</param>
          /// <param name="point2">The second point</param>
          /// <returns>The distance</returns>
          private float Distance(Point point1, Point point2)
          {
                return (float)Math.Sqrt(Math.Pow(point2.X-
point1.X,2)+Math.Pow(point2.Y-point1.Y,2));
          }

          /// <summary>
          /// Prints all the pda's and where they are located in.
          /// </summary>
          public void ToTheString(){
                for (int x=0;x<sizearrayx;x++) {
                      for (int y=0;y<sizearrayy;y++) {

                            Console.Write("[{0}][{1}] : Size ",x,y);
```

```
                              Console.WriteLine(" ({0},{1})-
      ({2},{3})",m_treeNodesArray[x,y].m_area.X,m_treeNodesArray[x,y].m_area
      .Y,m_treeNodesArray[x,y].m_area.Right,m_treeNodesArray[x,y].m_area.Bot
      tom);

                              if (m_treeNodesArray[x,y].m_elements.Count !=
                                    0) {
                                    Console.Write("Contains :
                                                "+m_treeNodesArray[x,y
                                                ].m_elements.Count+"
                                                elements.");
                                    for (int c=0;c<m_treeNodesArray[x,y].
                                          m_elements.Count;c++) {
                                          Console.Write( "(ID: "+
      ((PDA)m_treeNodesArray[x,y].m_elements[c]).ID + "|Location: " +
      ((PDA)m_treeNodesArray[x,y].m_elements[c]).Location.ToString() +
      "|Range: " +
      ((PDA)m_treeNodesArray[x,y].m_elements[c]).TransmissionRange + "),");
                                    }
                                    Console.WriteLine();
                              }
                        }
                        Console.WriteLine();
                  }
            }
      }
}
```

Appendix D : Dump output Results

Bibliography

[ARA] Mesut Güneş, Udo Sorges and Imed Bouazizi. ARA – The Ant-Colony Based Routing Algorithm for MANETs http://citeseer.ist.psu.edu/cache/papers/cs/27039/http:zSzzSzwww-i4.informatik.rwth-aachen.dezSz~mesutzSzpaperzSz2002_ARA_IWAHN.pdf/ara-the-ant-colony.pdf/

[ARID] Yih-Chun Hu, Adrian Perrig from the Carnegie Mellon University and David B. Johnson from the Rice University, Ariadne: A Secure OnDemand Routing Protocol for Ad Hoc Networks http://www.ece.cmu.edu/~adrian/projects/secure-routing/ariadne.pdf

[BAT] TSG-RAN WG1 meeting #16, Packet Data Capacity, UE power consumption, optimization proposals http://www.3gpp.org/ftp/tsg_ran/WG1_RL1/TSGR1_16/Docs/PDFs/R1-00-1236.pdf

[BOT] B. Tatomir BSc., http://www.kbs.twi.tudelft.nl/People/Staff/B.Tatomir/

[CBF] Holger Füßler, Jörg Widmer, Michael Käsemann, Martin Mauve and Hannes Hartenstein. Contention-Based Forwarding for Mobile Ad-Hoc Networks http://www.et2.tu-harburg.de/fleetnet/pdf/Fuessler2003b.pdf

[CVERSJ] Jared Miniman, Microsoft C# versus Java http://www.devhood.com/tutorials/tutorial_details.aspx?tutorial_id=76

[DYAB] Ronald Kroon, Dynamic vehicle routing using ant based control, http://www.kbs.twi.tudelft.nl/Publications/MSc/2002-Kroon-MSc.html

[DSR] David B. Johnson, David A. Maltz and Josh Broch. DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks http://www.eecs.harvard.edu/~mdw/course/cs263/fa03/papers/dsr-chapter00.pdf

[DNET] Getting .NET framework 1.1 http://msdn.microsoft.com/netframework/technologyinfo/howtoget/default.aspx

[ETJC] Extreme Tech, Java vs. C#, a Code-for-Code Comparison http://www.extremetech.com/article2/0,1558,1152079,00.asp

[IIN] Seoung-Bum Lee, Gahng-Seop Ahn, Xiaowei Zhang, and Andrew T. Campbell, INSIGNIA: An IP-Based Quality of Service Framework for Mobile ad Hoc Networks http://www.idealibrary.com

[MCE] John O'Donnell, Introduction to Pocket PC, http://www.c-sharpcorner.com/PocketPC/IntroPocketPCJOD.asp

[RA] Mesut Güneş, Otto Spaniol. Routing Algorithms for Mobile Multi-Hop Ad-Hoc Networks http://saturn.acad.bg/bis/pdfs/02_doklad.pdf

[XDD] Microsoft, XML Documentation Tutorial, http://msdn.microsoft.com/library/default.asp?url=/library/en-us/csref/html/vcwlkxmldocumentationtutorial.asp

[Vogt] Vogt, C. 1999. Creating Long Documents using Microsoft Word. Published on the Web at the University of Waterloo.

[ZRP] Zygmunt J. Haas, Marc R. Pearlman. The Performance of Query Control Schemes for the Zone Routing Protocol http://wnl.ece.cornell.edu/Publications/ton8.pdf