J.L. Boehlé

Digitally signed by J.L.Boehlé
DN: CN = J.L. Boehlé, C = NL
Reason: I am the author of this document
Date: 2004.06.13 21:15:01 +02'00'

'A COMMUNICATION LAYER FOR DISTRIBUTED DECISION MAKING'

by

J.L.Boehlé

A thesis submitted in partial fulfilment of
the requirements for the degree of

Bachelor of Computer Science

Rotterdam University

2004

‘A COMMUNICATION LAYER FOR
DISTRIBUTED DECISION MAKING’

by

J.L.Boehlé

A thesis submitted in partial fulfilment of
the requirements for the degree of

Bachelor of Computer Science

Rotterdam University

2004

Approved by _____
Chairperson of Supervisory Committee

_____

_____

_____

Program Authorized
to Offer Degree _____

Date _____

Rotterdam University

Abstract

'A COMMUNICATION LAYER FOR DISTRIBUTED DESCISION MAKING'

by J.L.Boehlé

Chairperson of the Supervisory Committee: Ir. M.M.M. Abdelghany
Department of Computer Science

A driver travelling from point A to B in a city constantly gathers valuable information about the environment and traffic conditions. This information remains with the driver and therefore precious information that in theory could be useful to other drivers is lost. By equipping automobiles with a PDA drivers are able, with the use of those PDA's, to form mobile ad-hoc communication networks, which enables them to exchange information.

This thesis addresses the problems related to creating and maintaining a dynamic wireless ad hoc network in a city street network. It discusses a simulation environment that simulates the creation, operation and maintenance of a dynamic wireless ad hoc network. The simulation should allow the users to exchange messages and files between one or multiple nodes present in the city street network in order to evade traffic jams and hazardous situations or to warn emergency services.

The conclusions draw from this report indicate the simulation environment is able simulate the workings of a wireless mobile ad hoc network in a realistic way although some hardware limitations prevent the simulation form simulating more than thirty concurrently active nodes.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The author wishes to express sincere appreciation to Ir. M.M.M Abdelghany and Dr. Drs L.J.M. Rothkrantz. Thanks to Ir. M.M.M Abdelghany for recommending the TU Delft, also for his interest, ideas, recommendations and input during his visits throughout the course of this bachelor thesis project. Special thanks go to Dr. Drs. L.J.M Rothkrantz for providing a most interesting bachelor thesis assignment, for his ideas and guidance during the weekly meetings and for taking time out of his busy schedule to facilitate this all.

Thanks also go out to B.Sc. B.Tatomir for his help with the traffic simulation program and his input during the weekly meeting as well as to the staff and students of the KBS group for making my stay a pleasant one. And Last but certainly not least thanks to my family for the support and advice given in relation to this bachelor thesis project.

GLOSSARY

ACK. Abbreviation of Acknowledgement, packet defined in the IEEE 802.11B specification indicating that the host has successfully received the data packet transmitted to it.

AHS. Abbreviation of Ad Hoc Simulation, program designed for this thesis assignment that simulates the behaviour of PDA's and mobile communication in a mobile environment.

AHV. Abbreviation of Ad Hoc Visualization, program designed for this thesis assignment that visualises the actions performed by the AHS program.

BANT. Abbreviation of Backward Ant, Agent part of the ARA Routing algorithm that is send in reply to a FANT to backtrack the route established by the FANT in order to create a bidirectional communications path.

CRC. Abbreviation of Cyclic Redundancy Check, common technique for detecting data transmission errors.

CPU. Abbreviation of Central Processing Unit, hardware specialized in calculations. The CPU is the most important part of a computer.

CTS. Abbreviation of Clear To Send, packet defined in the IEEE 802.11B specification indicating that that host is ready to receive incoming data.

FANT. Abbreviation of Forward Ant, Agent part of the ARA routing algorithm that is send by a querying node in order to establish a route to a specific destination node.

GUI. Abbreviation of Graphical User Interface.

IEEE. Abbreviation of Institute of Electrical and Electronics Engineers, United States based non-profit organization for the creation of standards for computer formats and devices.

MAC. Abbreviation of Media Access Layer, one of the two layers that make up the Data link layer of the OSI model, responsible for moving packets from an to a NIC.

MANET. Abbreviation of mobile ad hoc network.

Namespace. A logical grouping of the names within a program. The Microsoft Framework .NET refers to a namespace as a collection of classes that together from a library.

NIC. Abbreviation of Network Interface Card, device that enables a computer to communicate over a specified network.

PDA. Abbreviation of Personal Digital Assistant, handheld device that combines computing, telephone/fax, Internet and networking services.

RTS. Abbreviation of Ready To Send, packet defined in the IEEE 802.11B specification that is send by a host that wants to initiate data transfer with another node.

TCP/IP. Abbreviation of Transmission Control Protocol / Internet protocol, suite of communications protocols used to connect to hosts on the Internet.

Wi-Fi. Abbreviation of Wireless-Fidelity, comprises a set of standards for wireless local area networks based on the IEEE 802.11 specifications.

WLAN. Abbreviation of Wireless Local Area Network, network that uses radio waves as carrier.

Chapter 1

INTRODUCTION

This chapter discusses the problem setting that lies at the foundation of the project 'A communication layer for distributed decision making'. Also the traffic simulation program and the project goals will be discussed.

1.1 Problem setting

The traffic situation in a city like Rotterdam is an ever-changing series of events. During the peak traffic hours between 8:00 - 9:00 and 17:00 - 19:00 congestions are rampant across the city. If an accident occurs emergency services need to be notified as soon as possible with up-to-date information concerning the current situation. Any automobiles travelling through that area need to be diverted by the police to prevent a major traffic jam.

A driver travelling from point A to B in a city constantly gathers valuable information about the environment and traffic conditions. This information remains with the driver and therefore precious information that in theory could be useful to other drivers is lost. By equipping automobiles with a PDA drivers are able, with the use of those PDA's, to form mobile ad-hoc communication networks, which enables them to exchange information (an automobile equipped with a PDA is from now on referred to as a node).

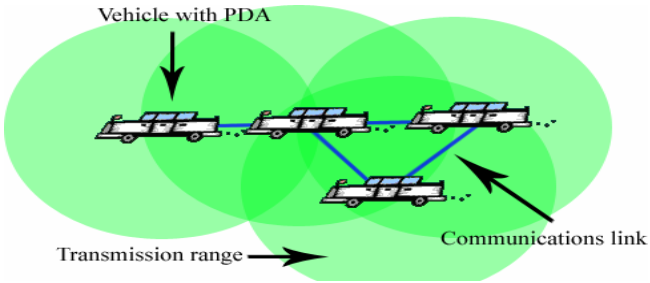

Figure 1: Nodes

Traffic is always on the move and mobile ad-hoc communication technology is only able to cover relatively small distances (550 meters/601.49 yards maximum). Dynamic self-organizing networks have to be formed in order to maintain communication with nodes even if they are out of the sending nodes transmission range. Nodes can enter and leave the network at any time; a node that leaves the network can do so without prior notice.

This thesis addresses the problems related to creating and maintaining a dynamic wireless ad hoc network in a city street network. It discusses a simulation environment that simulates the creation, operation and maintenance of a dynamic wireless ad hoc network. The simulation should allow the users to exchange messages and files between one or multiple nodes present in the city street network in order to evade traffic jams and hazardous situations or to warn emergency services.

1.2 Crisis management project

The project 'a communication layer for distributed decision making' is defined as a subproject of the 'Crisis management using mobile ad-hoc networks' **[R04]** project. The 'Crisis management using mobile ad-hoc networks' project aims to develop an environment wherein rescue services can communicate using handheld devices that can form dynamic mobile ad hoc networks. Through these handheld devices that operate in a wireless environment communication is still possible when major infrastructural communications links have been damaged, destroyed or overloaded. In case of a major disaster within a city, emergency service can communicate without the need for access points or other infrastructural requirements. The users of this network can exchange observations, message and files through agent technology and intuitive GUI's located on the handheld set. The agents help the user in finding, storing and retrieving information from the network.

The first implementation of the Crisis management project focuses on building a simulation environment that is able to model in a realistic way the behaviours of city traffic, mobile ad-hoc communication and agent technology.

Two other projects part, of the crisis management project that are strongly related to the project discussed in this thesis, namely the Traffic simulation program described in the following subsection and the Waypoint project. The Waypoint project defines information about the environment within the simulation environment in order to provide the user with information about his current location within the simulated environment.

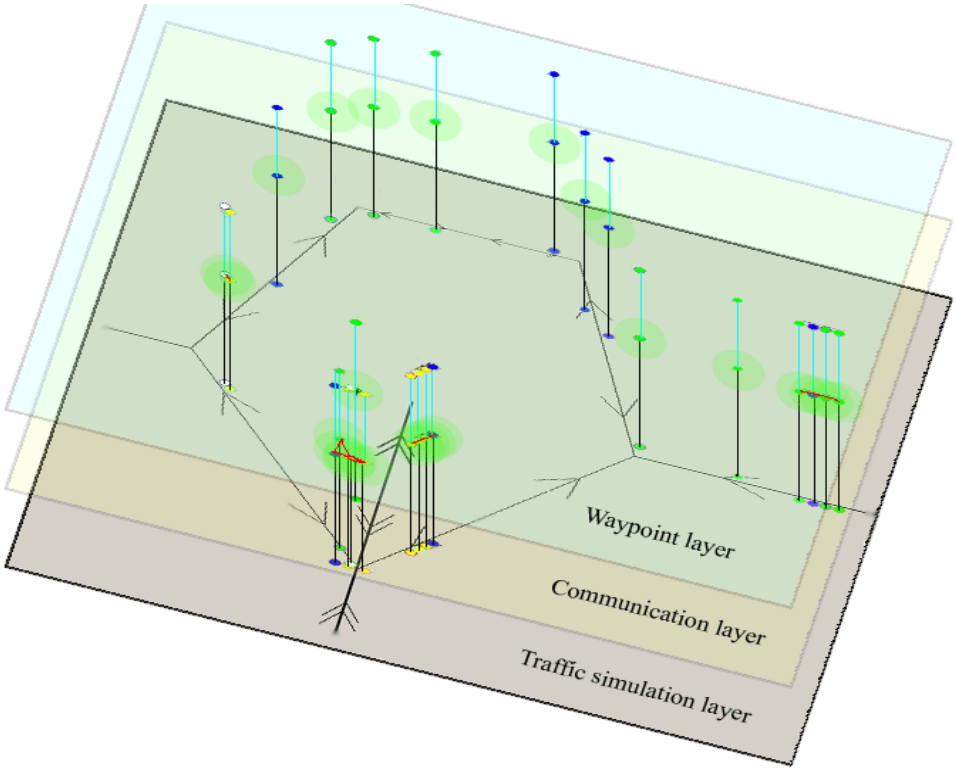

Figure 2: Crisismangement layers

The different layers in the Crisis management project implement specific features and are stacked on top of one another as shown in figure two. These layers should the exchange and provide information to the nodes so that drivers can interpret the information and take further action.

1.2.1 Traffic simulation program

The Traffic simulation program **[K02]** uses the Ant based control algorithm to route traffic through a city street network. The Ant based control algorithm calculates the fastest route, in time, for a car from point A to B.

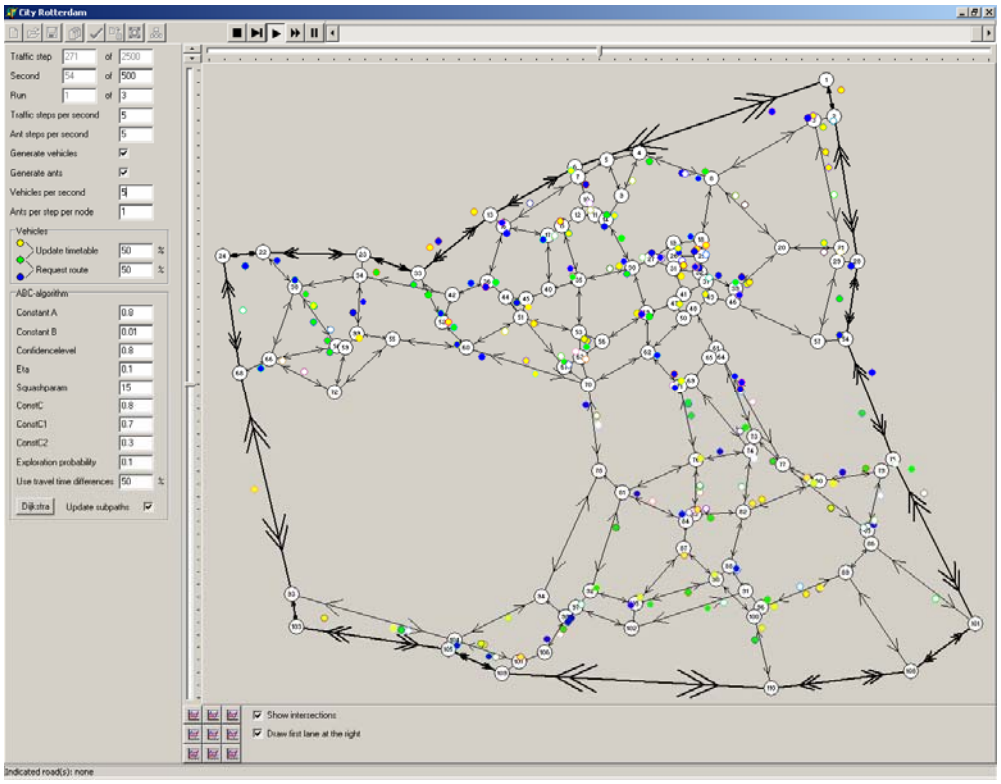

Figure 3: Traffic simulation program

With the use of traffic lights, precedence rules, roundabouts, one lane, multi lane and one-way roads a 'realistic' simulation environment is created. By recording the coordinate information of the cars during a simulation run in the traffic simulation program coordinate information data is gathered for the vehicles equipped with a PDA that run in the AHS/AHV simulation.

1.3 Project goals

The primary goal of this project is creating a communication layer that is able to create and maintain connections on a simulated ad-hoc dynamic network. A secondary goal in the creation of the communication layer is building a graphical user interface that displays the moving nodes and dynamic communication links. The tertiary goal is the realisation of a communication link with the traffic simulation program developed by R.Kroon in order to receive positional information about nodes.

Chapter 2

LITERATURE STUDY

This chapter summarizes the literature studied before starting the design and development of the simulation environment. Although more literature was studied concerning different routing protocols only the literature applicable to the final implementation has been listed here. The first section covers the specifics of the IEEE 802.11B MAC layer the second section discusses the ant-colony-based routing algorithm.

2.1 IEEE 802.11B MAC Layer

The IEEE 802.11B MAC Layer **[IEEE99a, IEEE99b]** consists of a series of agreements concerning the sending and receiving of data. In this chapter the agreements that will be implemented in the simulation environment are discussed. A table providing wait times and parameter descriptions can be found in appendix VIII. Appendix VI displays three flowcharts describing the flow of the overhearing process, packet transmission process and packet reception process for single and multi-fragment packets.

2.1.1 Probing

Before nodes, operating in an ad hoc wireless environment, can start to transmit data to each other they must first find each other. In order for a node to discover its neighbour nodes, which are within its communication range, the node sends out a probe.

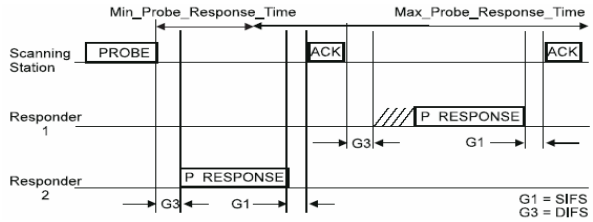

Figure 4: Station probing

Setting the type field of an IEEE 802.11B frame control structure to 'Management' and its subtype field to 'Probe' creates the probe message packet. Once created the message is broadcasted at 1Mbps in order to have the maximum transmission range. After transmission the node waits for a specified amount of time in order to receive all probe response messages. The probing node confirms each received probe response message with an ACK message.

A node that receives the probe message formulates a probe response message by setting the type field of an IEEE 802.11B frame control structure to 'Management' and its subtype field to 'Probe response'. After transmitting the packet to the probing nodes it waits a specified amount of time for the ACK message confirming the reception of the probe response packet. If no ACK message is received within the time period the probe message is retransmitted. This sequence is repeated up to four times.

2.1.2 Basic access method

In order to realize communication between two wireless IEEE 802.11B compatible nodes a complicated set of delay timers and packets is needed. Nodes operating in the wireless ad hoc mode cannot start sending data immediately. First the nodes must determine if the receiving node is currently idle before transmission can start.

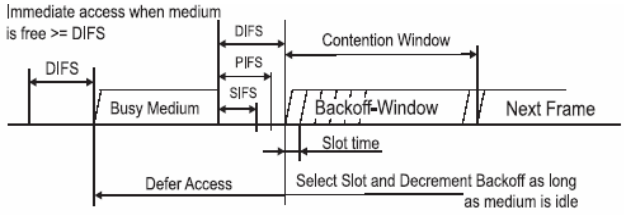

Figure 5: Basic access method

A node that needs to determine if the intended receiving node is idle, it opens a reception socket and senses the air for transmission that have the intended destination as destination. If no transmission is detected the packet is sent out to the destination. If the node senses a current transmission the node waits until the transmission has stopped. The waiting period is determined by initialising a timer, this 'back off' timer waits for a 'random' period plus the slot time for the current allocated for the current MAC layer (in this case 802.11B).

$$\text{Back off Time} = \text{Random}() * \text{aSlotTime}$$

The Random() function displayed here draws an integer value in the range of [0,CW] where CW is an integer value in the range between aCWmin and aCWmax, where $\text{aCWmin} \leq \text{CW} \leq \text{aCWmax}$. This value doubles every time a busy receiver is detected at the end of the back off period.

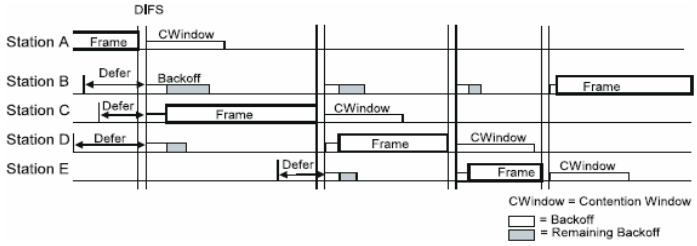

Figure 6: Back off procedure

This back off method is used in order to give every node a statistical equal opportunity to transmit its data to the destination. If there are no transmissions to the intended destination at the end of the contention period the node sends out its data.

2.1.2.1 Data transmission

The transmission of a data packet to a node starts when the sending has found the receiver to be idle. The sending node broadcasts an RTS packet to the intended destination then waits a certain amount of time for a CTS packet. If the CTS packet is not received within the waiting time the RTS packet is transmitted again, if the destination node is still idle. The CTS packet is send back using the broadcast method to notify other nodes of the pending transmission.

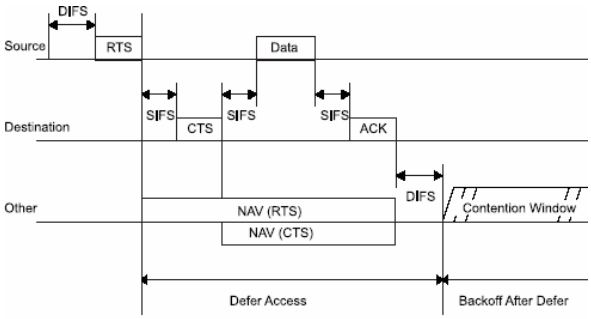

Figure 7: Data transmission communication

Nodes that receive either the broadcasted RTS or CTS message set an NAV timer specifying the length time in which the sending and receiving nodes are not available for transmissions. The time period for the NAV timer is taken form the duration field of the RTS or CTS packet. If no transmission is detected after the exchange of RTS or CTS a node may reset its NAV timer after a period specified by the formula below.

$$NAV\ reset = (2*aSIFSTime) + (CTS\_Time) + (2*aSlofTime)$$

After receiving the CTS packet and waiting for a SIFS amount of time the node can transmit the data packet to the destination. Once transmitted the node waits a specified time for the ACK packet to arrive confirming a good transmission.

2.1.2.2 Multi fragment

An 802.11B Data packet can have a maximum payload of 2312 bytes. Data that exceeds this size needs to be fragmented by the sender before transmission can take place. The 802.11B specification has limited the payload to 2312 bytes in order to minimize the bit error rate. The bit error rate in wireless network is at current 10^{-5} to 10^{-6}. Increasing the size of the payload will lead to an increased bit error rate, which eventually results in 100% packet loss.

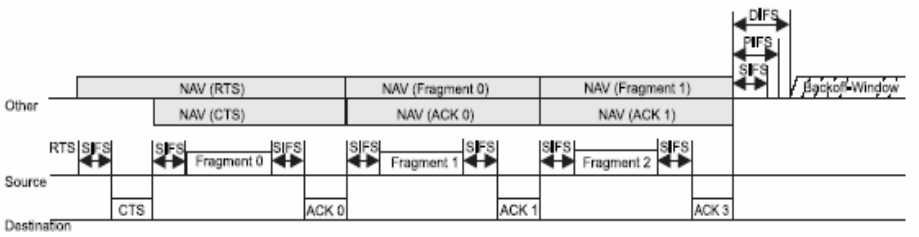

Figure 8: Multi fragment transmission

Once the sending node finds the destination node to be idle it sends out an RTS packet. The node that overhears this RTS packet sets it NAV timer for sender and destination using the duration field of the RTS packet, the destination node sends back a CTS. The sending node can now start to transmit its first fragment, nodes that overhear the fragment update their NAV timer for sender and destination, and the sender confirms the successful reception of the data packet using the ACK packet, which also may be overheard by other nodes. This process is repeated until the last fragment specifies zero for the more fragments field of the data packet.

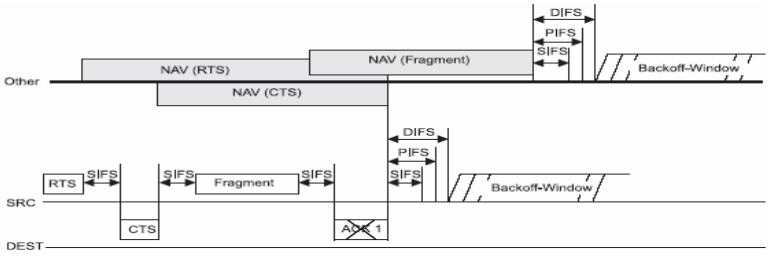

Figure 9: Missed acknowledgement.

If the ACK period expires on the sending node signalling a missed ACK packet the data transmission stops. The nodes that overheard the last fragment will have their NAV timer still running for the duration specified by the last fragment. The sending node may try to rebuild the communication with the destination node, if this does not happen the destination node will be available to all other nodes as soon as their NAV timers expire.

2.2 The Ant-colony-based routing algorithm for MANETs

The Ant-colony-based routing algorithm (ARA) **[GS02, GSB02]** is a multi-agent system in which ants form the individual agents. ARA is based on the behaviour that ants exhibit when searching for food.

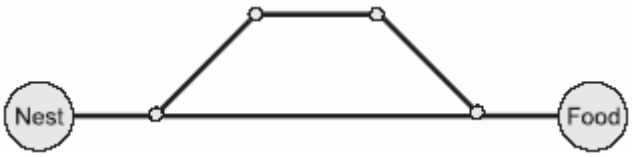

Figure 10: Two routes to a food source

Two ants travelling from the nest to a food source both take a different path. While proceeding along the chosen path they drop a pheromone that other ants can smell in order to trace back the taken route. Once an ant reaches the food source it picks up some food and starts travelling back along it previous path increasing the strength of the pheromone track. As displayed in figure ten the ant

that took the bottom route needs less time to travel to the food source and back therefore leaving a stronger pheromone track. Other ants in search of food will choose the stronger pheromone trail since this represents the shortest route to the food source in time.

2.2.1 The ARA algorithm

In order to make the ARA algorithm suitable for dynamic ad-hoc networks some adaptations have to be made. Since pheromone values cannot be dropped by current day communication devices a routing table is maintained on each individual communication device in the network.

Table 1: ARA routing table.

Destination address	Next hop	Pheromone

The routing table noted above shows the basic ARA routing table in which 'destination address' represents the address of the sending node, 'Next hop' represents the address of the previous node, which relayed the packet to the current node, and 'Pheromone' represents the strength of the trail.

The ARA algorithm consists of two formulas for increasing and decreasing the pheromone value.

$$\varphi_{i,j} := \varphi_{i,j} + \Delta\varphi$$

Figure 11: Pheromone updating formula

Each time a packet (ant) is received by a communication device (node) the routing tables are updated. The amount of pheromone φi,j for the route taken is updated with a constant value Δφ.

$$\varphi_{i,j} := (1 - q) \cdot \varphi_{i,j}, \qquad q \in (0, 1]$$

Figure 12: Pheromone decreasing formula

When operating in a dynamic network some routes may become invalid or are simply not used. In order to distinguish between optimal routes and sub optimal routes the amount of pheromone is decreased periodically for all routes. Pheromone values are decreased by the formula depicted in figure 12 where φi,j is the pheromone value and q is member of the set q ∈ (0,1].

2.2.2 ARA phases

The ARA routing algorithm consists of three distinct phases namely the route discovery phase, route maintenance and route failure handling.

2.2.2.1 Discovery phase

In order to communicate with other nodes the ARA protocol sends out a uniquely identifiable forward ant (FANT) that tries to establish a route to node somewhere in the network. At each node (hop) that it encounters on the way to the destination node it updates the route tables. If there is no address entry for the route taken by the FANT, the FANT senders address is noted is the 'destination address' field, the hop that relayed the packet is noted in the 'Next hop' field and the pheromone value is set to default. Otherwise only the pheromone value is updated using formula depicted in figure 11.

As soon as the FANT finds/arrives at the destination node a uniquely identifiable backward ant is launched (BANT). The BANT traces back the route to originating FANT node by making use of the routing table entries previously made by the FANT. While tracing back the route to the BANT updates the routing tables at each hop in the same fashion the FANT did. Once the FANT originator node is reached communication between the nodes can be established.

2.2.2.2 Route maintenance

The individual nodes do the route maintenance by analyzing received packets and by use of pheromone formula depicted in figure 12. Each packet received by the node either for processing or relaying is used to update the routing table and the pheromone value by use of formula depicted in figure 11. A timer determines the when formula in figure 12 is executed in order to decrease the pheromone value. Once the pheromone value for a certain route has reached zero the route can safely be discarded.

2.2.2.3 Route failure

The MAC layer implemented on the NIC of the communication device notifies the ARA protocol in case of route failure by returning a ROUTE_ERROR message. Once notified by the MAC layer the ARA protocol searches the routing table for an alternative route. If no alternative route is found the packet is relayed to all the neighbours of the node who will check their routing tables for a route to the destination. If the packet cannot be relayed to the destination node the packet is send back to the destination node, each receiving hop tries to find an alternative route as described above before sending the packet back.

Chapter 3

DESIGN

In this chapter the design of ad hoc simulation and ad hoc visualization programs will be discussed. The main considerations and design goals and features will be discussed as well as the UML Package and class diagrams created for the AHS and AHV applications.

The simulation environment consists out of three different applications namely Convert IT, AHS and AHV. The Convert IT program is a dataset creation tool and is described in detail in appendix I. The AHS program contains the entire simulation environment, it models the PDA's and communication layers and simulates the transmission of information over a dynamic wireless ad hoc network. The AHV program visualizes the actions that occur in the simulation environment. It displays the moving nodes, the possible network connections that can exist between nodes when they are in range and lists statistics that show the amount of traffic in the network.

3.1 Ad Hoc Simulation

When designing the simulation environment for wireless communication a number of requirements almost immediately arise. Wireless communication is limited by data transfer rates, transmission ranges and signal interference. PDA's are limited in resources like battery power, memory and processor speed. In this one hundred day project it is unrealistic to incorporate al these features to perfection. Therefore choices have to be made about the feature set to incorporate in the simulation environment the enables user to gain meaningful and useful results. The focus in the AHS program lies therefore completely upon

the communication layer and does not implement features like signal interference and PDA limitations.

Timing is an important factor when developing a simulation. In order to develop a realistic environment the simulation should be able to simulate data transmission in real-time. The IEEE 802.11B specification defines real-time as microseconds. By making use of the Windows WIN32 API a class was developed to access the high performance counter that registers time using the clock pulse generator of the PC, allowing for a time interval of 1/3579545. This should in theory provide for very accurate timing results but the retrieval of the frequency when ran in a multi–threaded environment under heavy load becomes unpredictable. Timing results start to vary and timers do not precise or are unable to report back of the short spans of time. Therefore this realism factor had to be abandoned. The times defined in the IEEE802.11B specification where scaled up to milliseconds in order to prevent varying timing results.

The package diagram depicted in figure 13 shows the different namespaces and the dependencies between the namespaces. The Communication layer packages are discussed in detail in section 3.1.1 the other packages will be discussed here. A detailed class diagram as well as a class dictionary can be found in appendix IV.

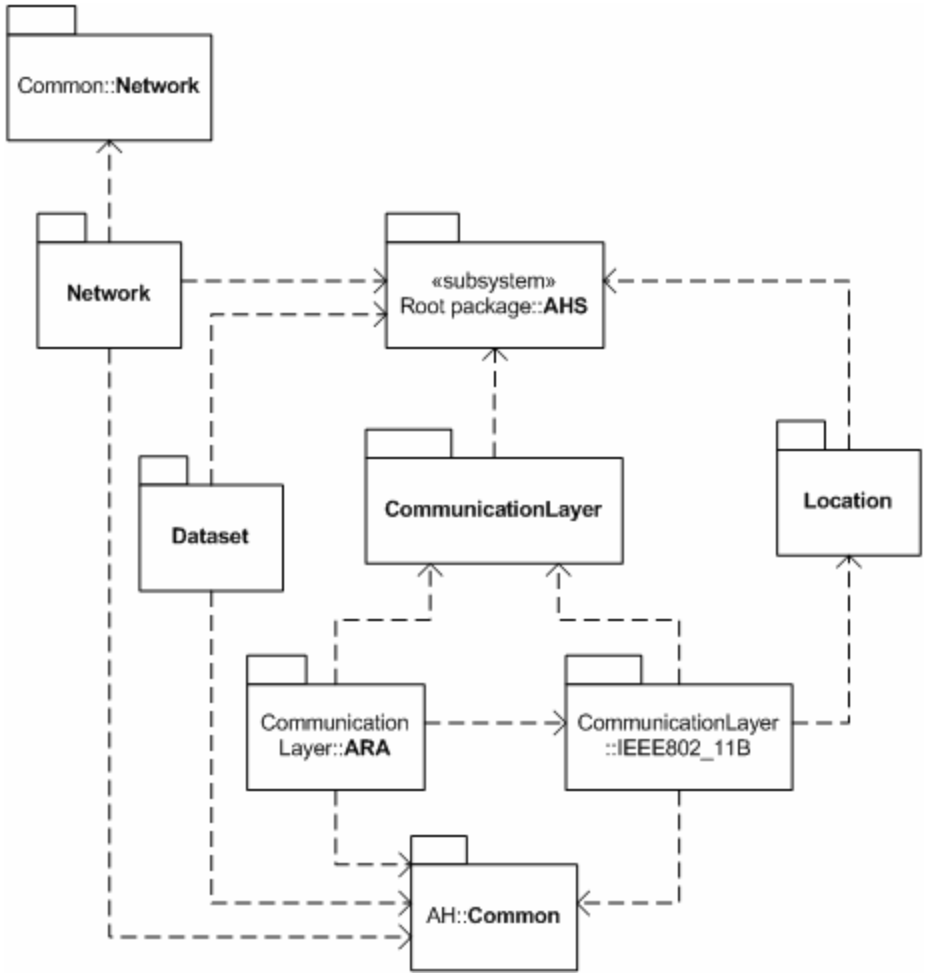

Figure 13: AHS packages

The AHS namespace contains the basic program classes like GUI's and the main simulation manager that controls the start-up and shutdown of the simulation environment. The Dataset namespace contains a dataset reader that reads traffic coordinate information form a dataset, further information about dataset creation and layout can be found in appendix I. The network namespaces contains functionality that provides a TCP/IP socket connection for communication between the AHS and AHV program. The location namespace contains the nearest neighbour detection algorithm. This algorithm is needed to determine

which nodes are within communication range in the simulation environment. This is described in detail in appendix III. The namespaces belonging to the Common namespace provide as a set of common features that is used by the AHS and AHV programs.

3.1.1 Communication layer

The communication encompasses the entire set of namespaces and classes that are together responsible for data transmission. The communication layer class diagram as depicted in figure 14 contains the classes that are part of the Communication layer namespace.

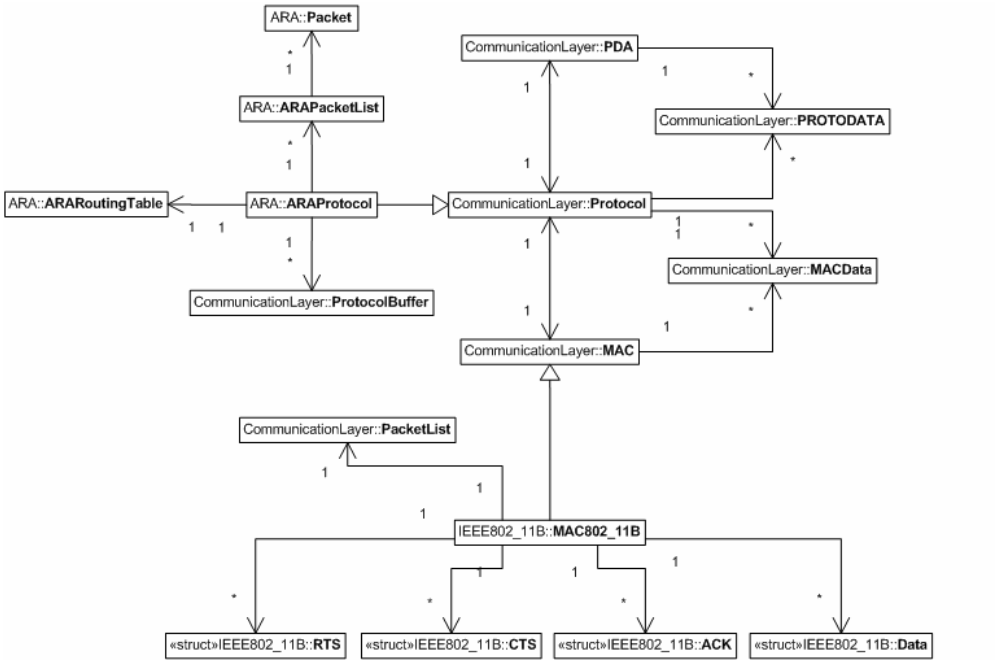

Figure 14: Communication layer class diagram

The Communication layer encompasses the functionality of an entire PDA, although only a sub set of features have been implemented. The PDA provides the files or messages that have to be transmitted, they travel through the protocol layer that fragments the data in the correct packet sizes and discovers and

maintains routes to nodes. The MAC layer then handles the IEEE 802.11B compliant packet forwarding and transmission of data to other nodes. The next sub-paragraphs will describe the design of the MAC and Protocol layers in detail.

3.1.1.1 Media access communications layer

Since realism was one of the most important requirements for the simulation environment a realistic wireless communication scheme had to implemented. The most widely used communication specification for PDA's is the IEEE 802.11B specification, part of the IEEE 802.11 set specifications also known as Wi-Fi. The working of the IEEE 802.11B specification is discussed in detail in chapter two section one, therefore this section focuses on the design considerations and portions of the specification implemented.

The IEEE 802.11B specification does not only cover the MAC layer but also discusses the requirements for the physical layer. The physical layer specification that covers the requirements for the radio signal at the electronic and mechanical level and is responsible for transferring the actual bit stream is not implemented in the simulation. Packets in the simulation will be passed from MAC layer to MAC layer in the MAC packet form. The transmission times for this layer are taken into account during the transmission of data.

The IEEE 802.11B specification defines a number of packets that can be transmitted via the MAC layer. These packets fall in three different categories namely control, management and data. The packets that fall in the control category are not considered in the simulation environment since they do not influence normal data traffic between mobile nodes.

The packets transmitted by the MAC layer although not converted to a bit stream are fully compliant with the IEEE 802.11B specification with the exception of the CRC number, this number is fixed since the possibility of data corruption in the simulation environment is non existent.

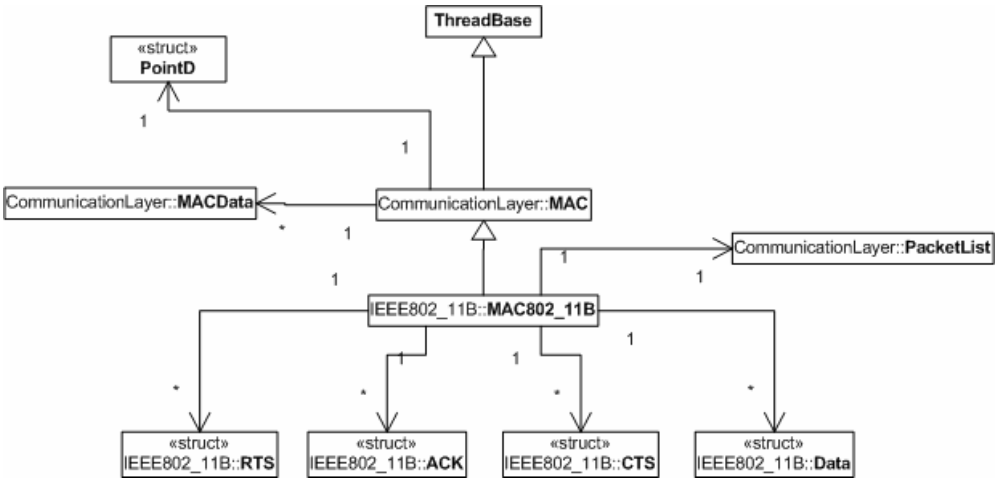

Figure 15: MAC layer class diagram

The design phase of this project resulted in the class diagram depicted in figure 15. The MAC Layer consists out of two MAC classes. The MAC class provides basic functionality that is needed to incorporate the MAC802_11B layer into the AHS simulation framework. The MAC802_11B class contains the communication functionality based on the IEEE 802.11B specification. The different packages implemented are depicted at the bottom of the class diagram. The MACData class provides a means of transferring data between the protocol and the MAC layer. The Packetlist filters out any duplicates received.

Appendix VI contains three flowcharts that depict the way in which a packet can travel through the MAC layer.

3.1.1.2 Protocol

The protocol incorporated in the simulation environment is specially designed to incorporate the ARA algorithm as described in chapter 2.2. ARA relies heavily on header data transmitted using the TCP/IP four specification. The packages transmitted by the protocol therefore contain the TCP/IP four fragment offset field, the other information required for the ARA protocol like Source address, destination address and previous transmitter is obtained from the MAC layer.

The protocol implemented in the simulation environment does not need to have an extensive feature set. Since the MAC layer described in the previous will handle the main bulk of the information exchange the protocol should only worry about data fragmentation, data buffering and Ant based route discovery and maintenance and duplicate Ant detection.

The routing table needed for the ARA algorithm controls the build up and flow of data over the network. In order to find and to communicate over the best possible data link the signal strength that is calculated by the MAC layer is incorporated into the pheromone update formula. The signal strength is calculated by the MAC layer as a percentage in relation to the distance between transmitter and receiver.

$$\varphi_{i,j} := \varphi_{i,j} + (\text{signal} / 100) + \Delta\varphi$$

The class diagram designed for the protocol is depicted in figure 16. The protocol is made up out of two classes namely Protocol and ARAProtocol. The protocol provides the basic functionality needed for the ARAProtocol class to be accepted by the framework. The ARAProtocol class incorporates the entire set of features needed for data transmission and ARA handling.

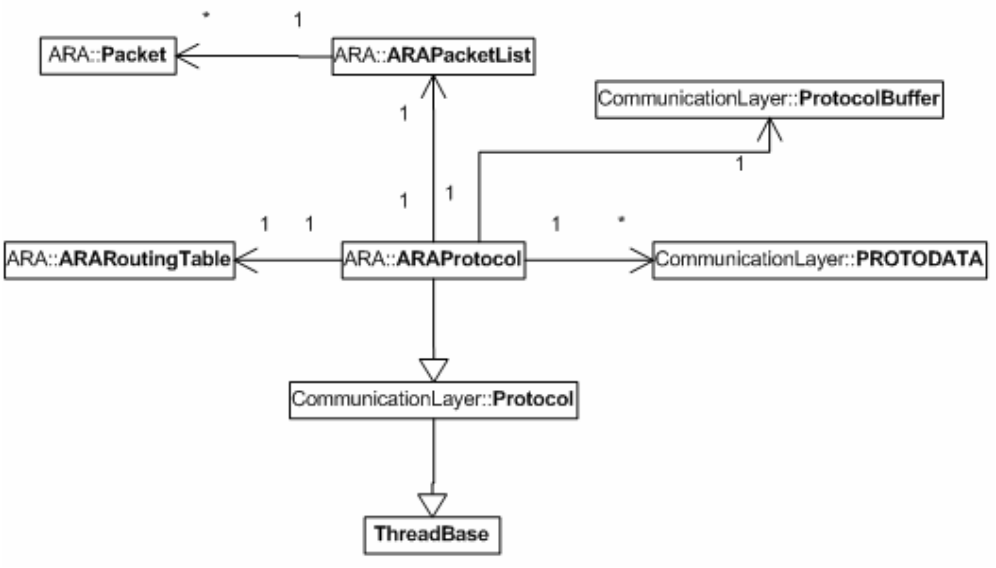

Figure 16: Protocol class diagram

The features include a packet list for filtering out duplicate data packets and ants. Since ants are broadcasted over the network in order to establish one or more routes to the host the reception of duplicate ants is quite common. The routing table that functions as small database in order to register, mutate and remove routes found by the host. The protocol buffer provides a means to store fragmented data transmissions received from one or multiple hosts and concatenates this stream to the final end result being a file or text message.

Appendix VII contains four flow charts that depict the way in which packets are send to other hosts and received by a protocol.

3.2 Ad Hoc visualization

The Ad Hoc Visualization environment is designed as an extension layer that operates on top of the Ad hoc simulation program. It uses a network connection to communicate with the AHS program, this set up allows for a distributed execution of the program.

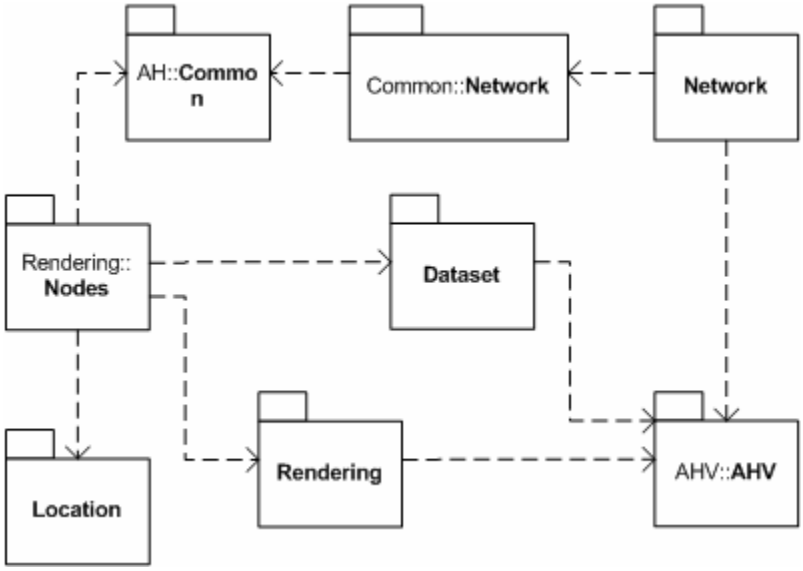

Figure 17: AHV packages

The packages displayed in the package diagram show the collection of namespace that form the AHV program. The class diagram of the AHV program can be found in appendix V as well as class dictionary.

The AHV namespace contains the collection of classes that provide the GUI's and control the simulation environment ran on the AHV program. The Network namespaces maintains a TCP/IP socket connection with the AHS, The presence of the connection between the AHV and AHV program is mandatory when running the AHV program.

The Dataset namespace provides functionality that reads nodes coordinate information from a dataset file, this class is almost identical to the AHS Dataset class. The location namespace contains the Nearest Neighbour algorithm that enables the AHV program to draw network connections between nodes with no data traffic over the network.

The rendering namespace contains a memory buffered rendering engine that enables the AHV program to provide smooth graphics without overdraw. Part of this render engine is the traffic map class that render the street network in the same manner as the traffic simulation program. The Nodes namespace part of the Rendering namespace contains the collection of active nodes.

Chapter 4

IMPLEMENTATION

This chapter focuses on the implementation details of the programs developed for the project 'a communication layer for distributed decision making'.

The entire project was implemented using the Microsoft C# language which is part of the Microsoft Framework .NET. This language is chosen because it provides the programmer with a 'managed' environment that does not 'leak' memory. The language is comparable by syntax to C/C++ and Java and therefore easy to learn. The Microsoft .NET framework allows for the incorporation of multiple languages like Managed C++, C#, J#, Visual basic and Delphi into one project, which eliminates any deficiencies that might exist between different programming languages, used in previous and future projects.

4.1 Ad Hoc simulation

The AHS program is a multi-threaded application that is responsible for the entire simulation. The AHS program reads node information from a dataset, calculates nearest neighbours and maintains the collection of PDA's, MAC layers and ARA Protocols. The program provides a minimal interface exposing only access to PDA statistics.

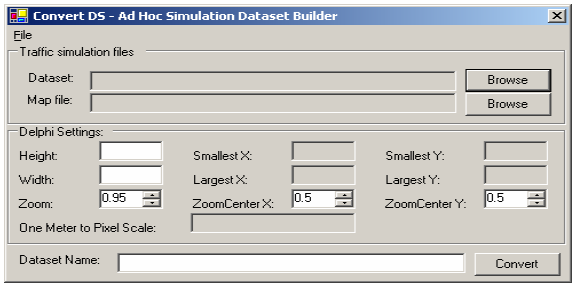

Figure 18: Convert IT program GUI

4.1.1 Communication Layer

This section focuses on the communication between the different classes in the communication layer. The internal communication of the MAC layer and Protocol layer classes will be discussed in the following paragraphs.

The Protocol class and the MAC class that operate within de communication layer both have their own thread. This allows the protocol and MAC layer to function independently form each other allowing data processing by the protocol and data transmission and reception by the MAC layer at the same time. This multi-threaded set up allows for an increased realism factor but at the same time complicates the program flow. Different threads cannot access data of other threads without proper synchronization mechanics, otherwise data corruption may occur. When implementing synchronization for threads the programmer always has to check that the program cannot deadlock. A deadlock occurs when two or more threads are waiting on each other's synchronized resources to become available again.

```
public bool Receive(Object packet,
                    int signalstrength,
                    TRANSMISSIONSPEEDS_KBPS kbps)
{
   ....

   if (!Monitor.TryEnter(_InputQueue))
    return false;

   ....

   _InputQueue.Enqueue(packet);
   Interlocked.Increment(ref _EnqueuedPacketsIn);
   Monitor.Exit(_InputQueue);
   ExitWait();

   return true;
}
```

Figure 19: Synchronized receive function

The C# source code depicted in figure 19 shows a part of the MAC802_11B class. Other MAC802_11B classes within the simulation environment access the Receive function when they want to send a packet to the host. The code demonstrates how thread synchronization is implemented. Multiple threads can access the Receive function. The first thread that reaches the Monitor.TryEnter statement acquires a lock on the incoming packets queue, other threads that attempt the gain the lock receive are diverted out of the Receive function with a 'false' value indicating sending failed.

This prevents a possible deadlock situation. As soon as the thread that acquired the lock has added its packet to the incoming packets queue it increments the incoming packet counter and releases the lock, then it wakes up the thread by calling the ExitWait function.

During the simulation PDA's get added, updated and removed from the PDAContainer. A PDA that gets registered as active in the simulation environment starts in sleep mode. This means that the threads for the PDA that encompasses the MAC and Protocol classes are set in block mode and do not consume any CPU resources, for further details see section three.

The MAC and Protocol classes almost have an identical build of packet reception and processing. If a PDA wants to transmit a packet, it creates a PROTODATA packet. This PROTODATA packet can hold numbers of parameters namely a filename, destination and a BinaryReader class that provides binary access to the stream of data that needs to be transmitted.

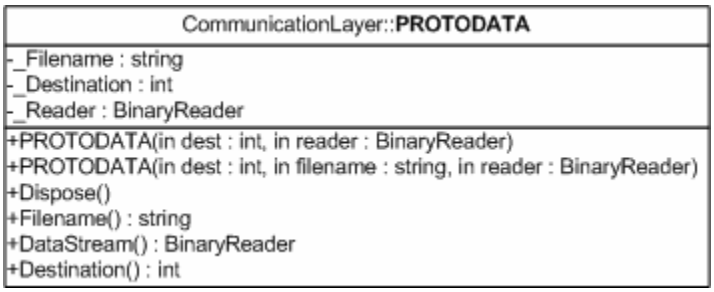

Figure 20: PROTODATA class layout

This packet is placed in the incoming queue of the protocol that will fragment and process the data before transmitting it to the MAC layer using a MACData class.

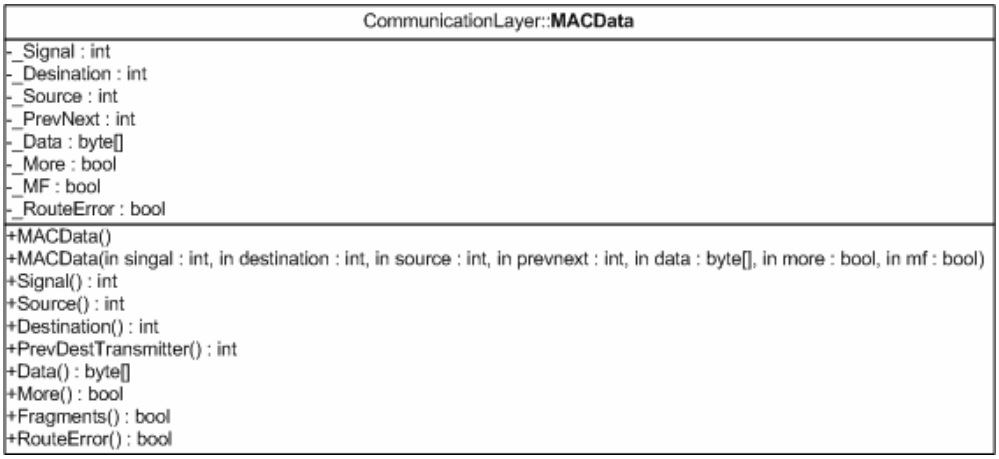

Figure 21: MACData class layout

The MACData class is used by both the MAC and Protocol classes to exchange data between the two classes. A protocol that transmits data to a MAC layer fills out the Destination, PrevNext, Data, More and more fragments (MF) field. A MAC layer that transfers data to a Protocol fills out all the fields with data received.

4.1.1.1 IEEE 802.11B media access communications layer

The MAC layer implemented in the simulation environment can reside in six different states. The MAC can be IDLE meaning that the thread has no work to carry out and currently resides in sleep mode. The MAC can be in the SIFS_WAIT state indicating the MAC is delaying transmission to allow the SIFS period to pass. The MAC can be in NAV_WAIT state indicating that the NAV timer has been set, the MAC will defer from transmitting data until the timer expirers or is reset. The MAC can be CTS_WAIT state this means that an RTS packet has been send to a host and the MAC layer is waiting for the CTS response. The MAC can be in DATA_WAIT indicating that the CTS packet has been transmitted and that data is expected within the duration period. ACK_WAIT meaning that data has been transmitted and the MAC is waiting for

an ACK packet reply. The six states mentioned above control the program flow and allow the MAC enter sleep mode once a packet has been send and pick up where it left a soon as a packet is received.

The hidden node problem (see figure 22) is one of the most common problems that occur in mobile dynamic ad hoc networks. The hidden node problem occurs when two or more nodes are within transmission of a node but not within each other's transmission range.

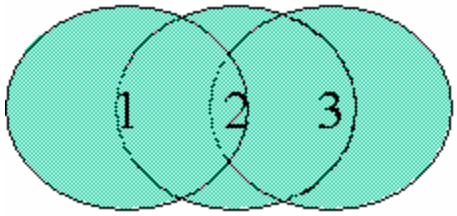

Figure 22: Hidden node problem

When node two is not transmitting any data node one and three are free to transmit data. Their now exists a possibility that node one and three start transmitting data to node two at exactly the same time. This results in packet collisions and prevents the reception of any data by node two.

The IEEE 802.11B specification solves this problem with three the use of the control packet types RTS, CTS and ACK. Also by defining in each packet send, using the MAC layer, a duration field. This duration field indicates maximum time available to transmit and entire sequence of packet to a host. This means that the RTS packet send by a host has a duration field value that is equal to the transmission time needed to transmit a CTS, Data, ACK packet and the waiting time of three SIFS intervals (10 μs). The source code depicted in figure 23 shows the formula used to calculate the duration time of an individual packet.

```
private int CalcPacketDuration(PACKETSIZE psize,
                               int datalenght)
{
  float TD_Data = sTP + sTPhy + (((8.0f * ((int)psize +
                  datalenght)) * 1000) /
                 (int)_TransmissionSpeed);

  // Round up to the next integer value
  if((TD_Data - (int)TD_Data) != 0)
      TD_Data++;

   return (int)TD_Data;
}
```

Figure 23: Packet transmission time formula

The function **[XR02]** requires the length of the physical layer preamble in bytes, the length of the physical layer header in bytes, the size of the packet header in bytes and the length of the data field if applicable, and finally the transmission speed in Kilobits per second with which the PDA is transmitting. The IEEE 802.11B specified values are listed in appendix VIII. The duration field is also used by node that overhears a packet transmission. The simulation environment allows nodes to overhear the control packets RTS, CTS and ACK. A packet that is destined for another node is used to update the NAV timer. The NAV timer specifies the amount of time, based on the duration time in the overheard packet, in which other nodes are communicating over the network and the current should defer from transmitting data.

To prevent all nodes form initiating a data transfer request at the same time each node must wait a random amount of time before starting to transmit data. The time interval period starts when the NAV timer has expired. The back off formula is implemented in the AHS program using the source code as depicted in figure 24.

```
private int Backoff()
{
   ....

   int backoff = sCWMin;

   for(int i = 0; i < _Retry; i++)
    backoff += backoff;

   if(backoff > sCWMax)
    backoff = sCWMax;

   return (backoff * sTSlot);
}
```

Figure 24: Back off algorithm

4.1.1.2 ARAProtocol

The ARAProtocol implemented in the simulation environment can reside in three different states that control the program flow. The IDLE state indicates the ARAProtocol class currently has no incoming or outgoing packets and is in thread sleep mode. The BANT_WAIT state indicates that the ARAProtocol has send out a FANT packet and is waiting for BANT packet to return or the timeout timer to expire. The ACK_WAIT state indicates that ARAProtocol has send out a data packet and is waiting for the ACK packet confirming successful reception of the packet.

The ARAProtocol uses a specially designed ARA packet filtering class. This class filters out packet based on source transmitter, previous transmitter and fragment offset number. The filtering of duplicate packets is an important feature in ARA since ants are broadcasted onto the network and propagated by other nodes using broadcasting.

Data transmissions received from other nodes are in buffered streams. The streams allow for inserting, updating, removing and positioning. The protocol buffer incorporated in the ARAProtocol class allows for backing up different types of data like files, messages and AHV destined messages. The fragments are stored in buffer and the buffer is automatically disposed when the transmission is complete. The buffer will than perform the action defined by the buffer, like storing data to disk or showing a message box conforming successful reception.

4.2 Ad Hoc visualization

The AHV program consist mainly out of GUIs and menus that are rather trivial to implement using drag and drop and therefore will not be discussed. This section will therefore only discuss the implementation of the render engine.

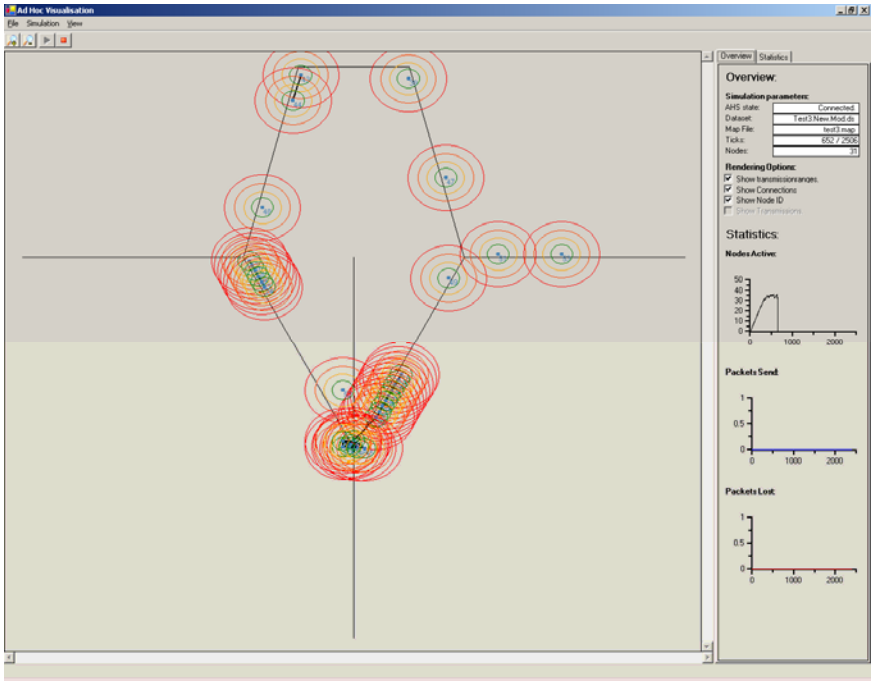

Figure 25: AHV program GUI

The windows graphic device interface (GDI+) that is responsible for drawing 2D graphics in the windows operating system. The render engine is required to make extensive of the GDI+ for drawing nodes, transmission ranges, city street network and network connections. The GDI+ provides extensive functionality and options for drawing that make it easy to use, the downside though is that de GDI+ is rather slow.

The traffic map used by the simulation program to route the traffic is rendered onto a bitmap image surface in the computers memory. Once rendered in memory the traffic maps bitmap image does not change, unless the user zooms in or out or scrolls using the scrollbars of the AHV program. This prevents unnecessary calls the GDI+ interface.

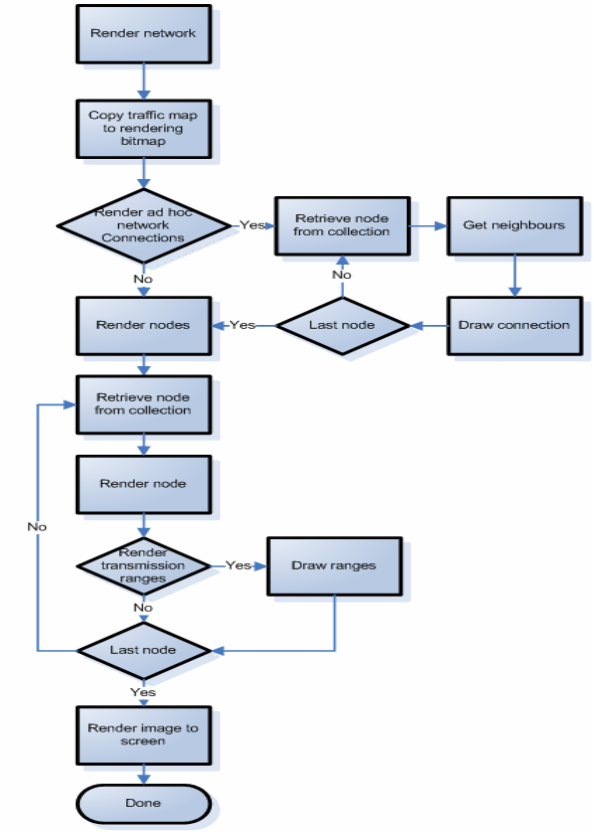

Figure 26: Render engine rendering flow chart

The rendering process of the rendering follows the path depicted in figure 26. The render engine uses a time interval for rendering the network to screen. The rendering process can be initiated by a number of events. It can happen after the expiration of the time interval, when the user zooms in or out, when the user modifies the position of the scroll bars and when the user resizes the form.

The rendering process starts by copying the traffic stored in memory to a rendering bitmap image surface. The rendering engine then checks to see if the network connection should be drawn, this is a user modifiable parameter. If so the connections are render to the rendering bitmap image surface. This process is done before render the nodes and transmission ranges to prevent connection

lines from drawing over nodes. After the connections are rendered the nodes and their transmission ranges are drawn on the rendering bitmap image surface. The rendering engine then presents the rendered bitmap image to screen.

The use of this double memory buffered rendering sequence prevents screen flickering caused by rapid overdraw when rendering the nodes or their connections.

```
public static int TransformToScreen( Double coord, int pMin,
                            int pMax, int pScreenSize,
                            double pZoom, double pZoomCenter )
{
   if (pMax <= pMin)
    pMax = pMin + 100;

   Double Temp = ( ( ( (coord- pMin) / (pMax - pMin) ) -
                   pZoomCenter ) * pZoom) + pZoomCenter;

   return Convert.ToInt32(Math.Round(Temp * pScreenSize));
}
```

Figure 27: TransformToScreen functions

In order to render the nodes and their connections the normalized coordinates have to be converted to screen coordinates. The transformation is carried by the function displayed in figure 27, and was taken from the traffic simulation program. It reveres operations done by the TransformToNorm function described in appendix I.

4.3 Common code base

The common code base found in the namespace AH contains classes that are used by the AHS and AHV program. These classes provide basic functionality that is used throughout the AHS and AHV program.

The Network namespace part of the common code base contains two classes that provide basic TCP/IP socket connection capabilities that can be inherited by other classes. The NetworkSocket class implements the Framework .NET TcpListner class and provides initialize and listen functions in order to start the socket server. The NetHandler class is a wrapper class around the NetworkSocket class and provides threading functionality for other classes to inherit.

The PointD structure represents a two-dimensional vector where the values for the x-axis and y-axis are stored in the double format. This means that coordinates can be stored with 15 ~ 16 decimal digits. This functionality is needed because of the fact that the traffic simulation program calculates all its coordinates using the double format.

The Threadbase class provides basic threading functionality for other classes to inherit. The threading functionality provides thread starting and stopping and also a thread wait mechanism. The thread wait mechanism revolves around blocking a thread by locking an object. As soon as we want to suspend a thread we let the thread lock a meaningless object that has been allocated only for locking purposes. The lock on the object blocks the thread for getting CPU time.

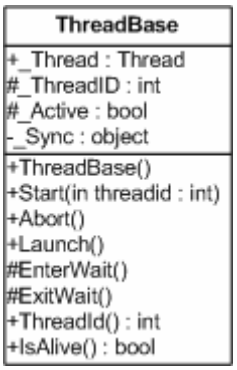

Figure 28: Thread base class

Once locked only the class that is locked can unlock itself. This prevents random unlocking of threads when their activity is not required. An example of this is the fact that in the simulation environment the MAC and Protocol layer only unlock themselves as long as they have data to transmit or process.

Chapter 5

EXPERIMENTS AND RESULTS

The experiments and results chapter covers two different sections. The first section will measure the performance of the AHS and AHV program. This will provide an overview of system memory consumption, CPU time consummation and network traffic generated between the AHS and AHV program. The second section in this chapter will measure simulation performance and behaviour.

The test system used to conduct the system was build up out of the following components: Intel Pentium four 3Ghz hyper-treading processor, 1024 Mb internal system memory running at a combined speed of 800 MHz, SATA 120 GB hard disk, and a ASUS PC4800-deluxe main board with integrated 1000 Mbps NIC.

5.1 Simulation environment

The simulation environment was run with the AHS and AHV container on one PC. The programs where loaded and prepared for a default simulation using the Test3.New.Mod.ds dataset. The AHS and AHV application where monitored over a 120 second period.

5.1.1 AHS

The AHS program ran for the first 120 ticks of the Test3.New.Mod.ds dataset, using the default parameters provided by the simulation environment. At the end there where 11 PDA's running represented by 22 threads. The entire program consisted of 27 threads.

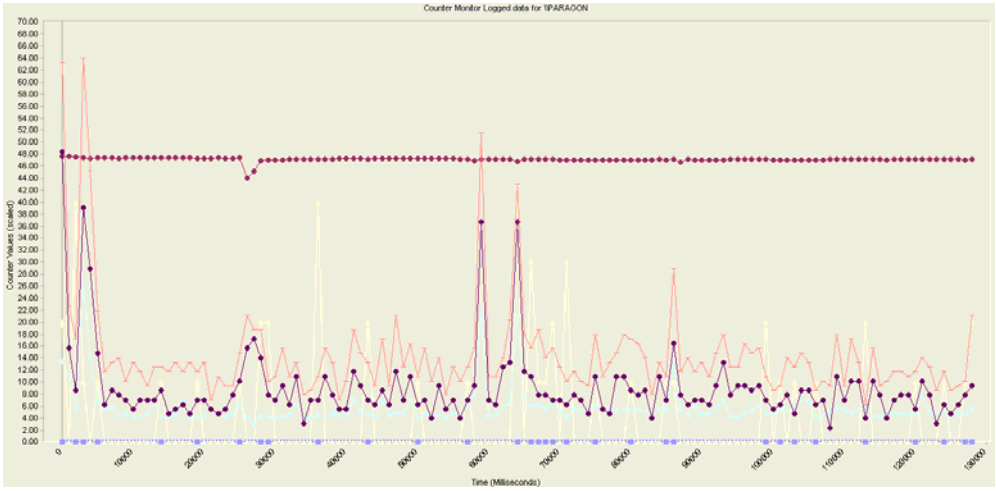

Figure 29: AHS counter monitor

The counter graph depicted in figure 29 show the behaviour of the AHS program in detail. The first peak at the start of the graph represents the program start up. The second peak is generated by the start up of the network socket. The simulation was stared about ten seconds after program start up. The first spike in the graph that then occurs is the registration of the first PDA and allocating of memory for the PDAContainer and the PDA, Protocol and MAC. The spike that occurs during the rest of the graph at 58, 64 and 85 seconds resemble increased network activity in the simulation. The increased activity was caused by text messages that where transmitted between nodes in the network. Figure 30 contains the legend associated with the counter monitor.

Counter Name	Graph Scale	Average	Min	Max	Std. Dev.
Redirector : Network Errors/sec	1.0	0.000	0.000	0.000	0.000
Memory : Available Bytes	0.0000001	471292256.000	439545856.000	476237824.000	3664702.000
System : Processor Queue Length	10.0	0.411	0.000	4.000	0.863
System : Context Switches/sec	0.001	5745.093	2929.000	26550.000	3151.826
Processor (_Total) : % Privileged Time	1.0	9.090	2.344	48.438	6.554
Processor (_Total) : % Processor Time	1.0	14.686	6.250	64.063	8.800

Figure 30: counter monitor legend

From the testing result gained we can conclude that the AHS application has an overall low utilization of the CPU, although this increases as soon as messages are exchanged between nodes within the network. The application benefits on the test system of the hyper threading functionality of the CPU, processors with out this capacity will show a slight increase processor time between five and fifteen percent according to Intel.

5.1.2 AHV

The AHS program ran for the first 120 ticks of the Test3.New.Mod.ds dataset. At the end of the simulation there where 11 nodes active, the entire program consisted out of 5 threads.

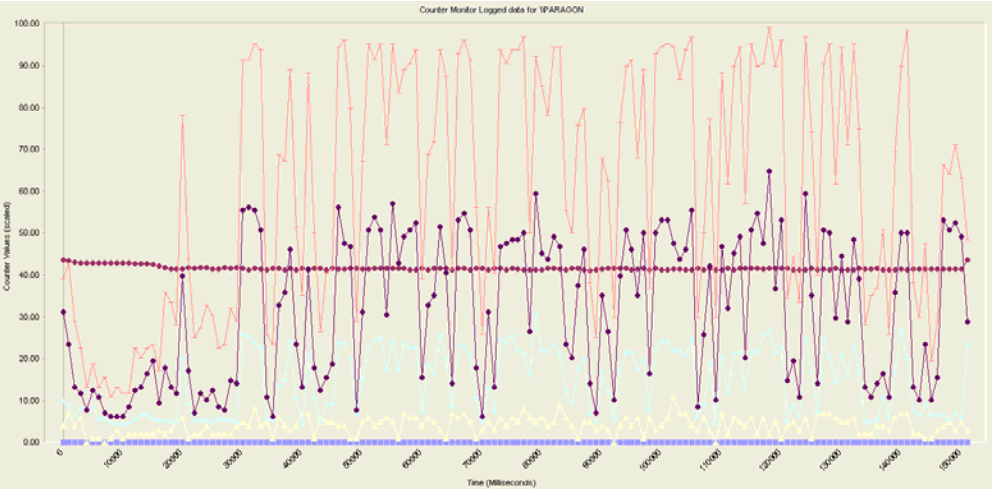

Figure 31: AHV counter monitor.

The counter monitor graph depicted in figure 31 shows that the AHV program starts to consume reasonable amounts of processor time each time the network layout is rendered. Figure 32 contains the legend associated with the counter monitor.

Counter Name	Graph Scale	Average	Min	Max	Std. Dev.
Redirector : Network Errors/sec	1.0	0.000	0.000	0.000	0.000
Memory : Available Bytes	0.0000001	416533696.000	411394048.000	435683328.000	5019904.000
System : Processor Queue Length	1.0	4.079	0.000	11.000	2.130
System : Context Switches/sec	0.001	14404.428	4324.000	30599.000	7789.882
Processor (_Total) : % Privileged Time	1.0	31.605	6.250	64.844	17.612
Processor (_Total) : % Processor Time	1.0	61.287	10.938	99.219	28.919

Figure 32: counter monitor legend.

In order to explain the behaviour of the AHS program and to find the source of processor time consummation a call graph was created. The AHV program was run under the identical settings used for the counter monitor. The call graph pin pointed the screen drawing calls made by the AHV render engine as the major time consuming operations. This behaviour can unfortunately not be altered since its Windows related.

5.2 Simulation experiments

Due to the lack of an agent framework that automatically sends out agents over the network the communication layer has to be operated by the user. This somewhat limits the extend to which we can stress test simulation engine. The results shown in this section should be repeated when an agent simulation framework is implemented in order to show the true network performance under heavy load.

The simulation environment it self does not show the working of the MAC and protocol classes to the user. To allow proper testing of the functionality a number of debug messages have been inserted into the MAC and protocol classes. These debug messages where used to generate the results listed in the following subsection and where removed when the testing was done since the hinder program flow.

5.2.1 MAC802_11B class

In order to test the MAC802_11B class a four nodes where defined that where all within each others transmission range. MAC class one was given the assignment to transmit a simple test message to MAC four which was located 549 meters (600.39 yards) away from node one, MAC two was located 159 meters (173.88 yards) away from MAC one and MAC three was located 269 meters (294.18 yards) away from MAC one.

The test was executed using the default IEEE 802.11B specified parameters and should produce the following result. MAC one should transmit an RTS with destination MAC four using the 11Mbps speed setting. Node two who is within range the 11Mbps signal should receive the RTS packet, process it and set its NAV to defer from sending. MAC one should experience a time-out since node four does not reply and should send the RTS frame again using the 5.5Mpbs speed setting. This should result in MAC two and three receiving the packet and update their NAV timer accordingly. MAC one experiences yet another time out and should scale down again to the 2Mbps speed. MAC two and three should receive the packet and update their NAV timers. MAC one should experience another time and decrease its transmission speed further to 1Mbps bringing node four into transmission range. MAC one transmits its packet and MAC two, three and four receive the packet. MAC two and three should update their NAV timers and MAC four should reply with a CTS packet directed back at node one. MAC two and three receive the packet and update their NAV timer and MAC one should processes the CTS packet and transmit a data packet. MAC four should receive the data packet and process it and reply with an ACK packet. MAC one, two and three should receive the ACK packet, MAC two and three should reset their NAV timer and all MAC layers should return to Idle.

The results of the test are listed in appendix IX. The test proved to be successful the MAC802_11B worked accordingly to the expected results which are in compliance with the IEEE 802.11B specification.

5.2.2 ARAProtocol

In order to the test the ARAProtocol class four PDA where setup in a network. The PDA's where placed on a imaginary line and setup so that the distance between a PDA and its neighbour was 549 meters (600.39 yards). PDA one was located at far left side of the network and PDA four was located at the far right side with node two and three in between them.

The MAC802_11B class was tested in the previous chapter and therefore we assume it functions correct. A number of debug message where inserted in the ARAProtocol class. The simulation was setup so that ARAProtocol one should transmit a file of 19 kb to node four. The expected results are as follows. The ARAProtocol one receives the send assignment and tries to find a route in its routing table to ARAProtocol four, the route cannot be found so ARAProtocol one should send out FANT message. The FANT should be received by ARAProtocol two who updates it routes and should send FANT the out again. The FANT is now received by ARAProtocol one and three, one discards the FANT since and three updates its routing table and sends the FANT out again. ARAProtocol two and four now receive the FANT, two should detect a duplicate packet and discard it, four should update its routing table and destroy the FANT packet, it then should create a BANT packet and send it out. ARAProtocol three then should receive the BANT packet, update its routing table and send the packet out again. ARAProtocol two and four should receive the BANT, four should detect a duplicate packet and discard it, two should update its routing table and send the packet out again. ARAProtocol one and three should then receive the BANT packet, three should detect a duplicate and discard the packet,

one should update its routing table and destroy the BANT. One then must build a data packet and send it to four using the newly detected route, after passing through ARAProtocol two and three, four should receive the data packet allocate a buffer and reply with an ACK packet. After reception of the ACK packet ARAProtocol should send the second data packet until there are no more packets.

The test results are listed in appendix X. The ARAProtocol was able to successfully establish a route between ARAProtocol one and four and was able to successfully transmit data over this link.

Chapter 6

EVALUATION AND RECOMMENDATIONS

This chapter contains the evaluation held at the end of the project and provides the reader with a number of recommendations concerning further improvements to the simulation environment, and the whole crisis management project.

6.1 Evaluation

The first weeks of the project focussed on research into Agent platforms, File distribution methods, wireless communication specifications and writing the plan of approach. At the end of that period the conclusion was drawn that building a simulation environment that included an agent platform, file distribution and wireless communication was not feasible within the given time period. This decision put to waste a lot information gathered about file distribution and agent platforms, but was necessary in order to make the project fit within the time limit.

The available time allocated for the design phase of three weeks was sufficient for designing a basic framework for the application although a number of revisions had to be made during the implementation phase. The revisions made where minor and concerned the implementation specifics of the nearest neighbour algorithm. Firstly a quad tree was implemented but this proved to error prone and slow. To design a framework that incorporated all the features needed would have taken the entire thesis period, therefore the design only focussed on the simulation layer.

The implementation phase of the project exceeded the allocated the time originally allocated by 15 workdays. This prolongation of the design period was

caused by delays that were encountered when developing the nearest neighbour algorithm and development of the Convert IT application. The integration of the AHS and AHV application with the Traffic simulation turned out to be problematic since the Traffic simulation program was not designed to provide coordinate information to other programs. The extraction and normalization of the coordinates made a direct coupling between the Traffic simulation and the AHS and AHV program complex and was not feasible.

At the end this of period the conclusion can be drawn that, although the simulation cannot handle more than thirty concurrent transmissions or active protocol and MAC threads due to hardware limitations, and that timing on current hardware is not able to be precise below the millisecond threshold, the project assignment requirements as defined at the start of the project and noted in chapter one have been met. The simulation environment allows for easy extensions and modification due to its namespace oriented design. The simulation environment allows nodes to form dynamic self-organizing networks and exchange data with a high degree of realism. The simulation environment functions by using the Traffic simulation generated data in the form of datasets.

6.2 Recommendations

The recommendations in this section reflect the sole opinions of the author concerning further development and improvements to the developed programs and the Communication layer and Crisis management projects, these opinions do not necessarily reflect those of other individuals involved in the projects 'a communication layer for distributed decision making' and 'Crisis management'.

6.2.1 A communication layer for distributed decision making

The keyword used during this simulation was 'realism' therefore large amounts of time have been invested to implement an IEEE 802.11B compatible MAC layer. Although the author believes this has succeeded there are other possibilities that

enable the simulation to have and IEEE 802.11B without the entire sequence of control packets. These packets can be replaced using formulas specified in **[XR02]** since IEEE 802.11B is very time depended and restricted a realistic simulation should not be impossible to achieve.

The ARA algorithm used in the protocol class is very depended on the TCP/IP four specification and is not compatible with the TCP/IP six specification since the fields of which ARA is depended like the 'fragment offset' have been removed in this version. TCP/IP six is not yet an issue but might become one very soon. Further research in the extendibility and revision of the ARA algorithm is needed to grantee a realistic simulation environment as time progresses.

The most obvious downside of the current simulation environment is the fact that it cannot influence the flow of traffic. Although this fact was clear from the start and agreed upon further integration between the traffic simulation and the simulation environment will further improve the realism factor and improve the way in which simulations can be setup.

The multithreaded implementation of the AHS program provides the user with a possibility to realistically simulate the communication between different nodes in the network. This however has certain limitations as described in chapter five. By implemented a more distributed orientated AHS program which allows for the transparent creation and operation of different nodes in different AHS subprograms divided over multiple PCs a greater network could then be simulated. This distributed system feature would further improve the realism factor of the simulation and will increase the testing capabilities of the network.

6.2.2 Crisis management

The 'Crisis management' project consists out of a number of independent programs developed by and altered by different students over different time-periods. Although this is normal for each development process it does in this case not benefit the compatibility between the different programs. The traffic simulation program has been developed in Borland Delphi five and seven, the waypoint server is developed in Delphi seven and the Communication layer designed described in this thesis is developed in C#.

The choice for C# could at first be considered a strange one but if we look at the current heading Delphi is moving in and the possibilities this might bring the choice is not so strange. The latest version of Delphi, version eight has fully migrated to the Microsoft Framework .NET of which C# is a main programming language. New Microsoft Windows version provide full support for the Framework .NET and their even exists a Linux version developed by Novell.

Taking all of the lesson learned in developing the different programs belonging to the Crisis management project a new component orientated framework could be designed and implemented in .NET that incorporates all programs. This framework could the form the base of further development.

BIBLIOGRAPHY

[IEEE99A] LAN MAN Standards Committee.
Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications.
IEEE, 1999.

[IEEE99B] LAN MAN Standards Committee.
Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band.
IEEE, 1999.

[K02] Kroon, Ronald.
Dynamic vehicle routing using ant based control.
TU Delft, 2002.

[GS02] Güneş, Mesut & Spaniol, Otto.
Routing Algorithms for mobile multi-hop ad-hoc networks.
International workshop NGNT, 2002.

[GSB02] Güneş, Mesut & Sorges Udo & Bouazizi, Imed
ARA – The Ant Colony Based routing algorithm for MANETs.
Aachen University of technology, 2002.

[P97] Brenner, Pablo.
A Technical tutorial on the IEEE 802.11 protocol.
Breeze COM, 1997.

[XR02] Xiao, yang & Roshdahl, Jon.
Throughput and delay limits of IEEE 802.11.
IEEE Communications Letters, 2002.

[R04] Drs. dr. L.J.M.Rothkrantz.
Crisis management using mobile ad-hoc networks.
TU Delft, 2004

Jupiter media Corporation.
Webopeida.
http://www.webopedia.com.

Microsoft Corporation.
Microsoft Developers Network (MSDN).
http://msdn.microsoft.com.

APPENDIX I

CONVERT IT

Although not part of the actual simulation environment discussed in this thesis, the Convert IT program is a vital tool for the creation of simulation datasets. The Convert IT program functions as a link between the traffic simulation software and the AHS and AHV programs.

The traffic simulation program only calculates the nodes coordinates during the screen-rendering phase of the program. The node coordinate information is discarded directly after the node has been drawn on the screen. Since this coordinate information is vital to the AHS and AHV simulation environment modifications have been made to the traffic simulation program in order to make the coordinate information persistent by writing it to a file.

```
------------------------------------------------------------
1 495 415  5.11000000000000E+0002  2.82558333333333E+0002
0 804 277  9.96450000000000E+0002  2.82558333333333E+0002
------------------------------------------------------------
```

Figure 33. Traffic simulation coordinate information

The coordinate information is stored by the traffic simulation program using the layout shown in figure 33. The beginning of every rendering sequence is marked with a line of '-' tokens. The lines between rendering sequences contain node information. The first parameter contains the nodes 'unique' number. The second parameter contains the x-axis screen coordinate for the node. The second parameter contains the y-axis screen coordinate for the node. The third and

fourth parameter contain the screen x and y-axis coordinates for the roadblock with full precision.

I.1 Design

The Convert IT program was designed in order to make traffic simulation data useable for the AHS and AHV program. The main goal when developing the application was automating the conversion process, transforming the traffic simulation screen coordinates back to traffic map file coordinates (normal coordinates) and removing duplicate entries for nodes made by the traffic simulation program.

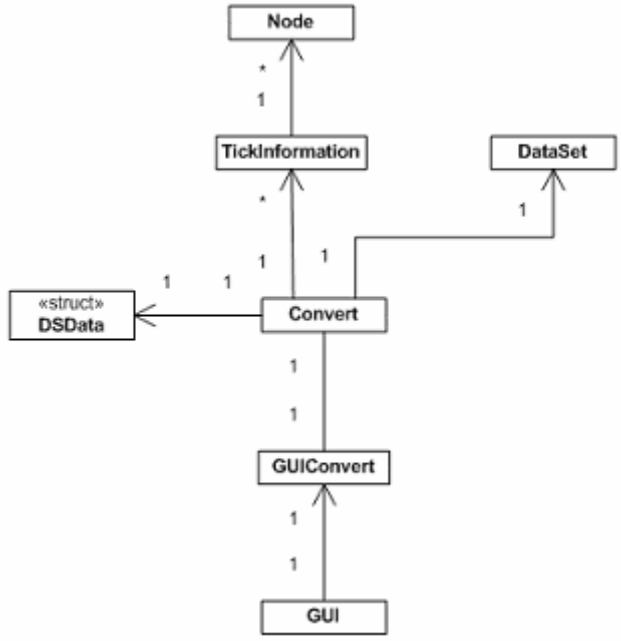

Figure 34: Convert IT design

The Convert IT program is a multi-threaded application and is designed accordingly to the class diagram shown in figure 34. Table two lists the class names and gives a brief description about the their purpose.

Table 2: Description of classes present in the Convert IT class diagram.

Class name	Description
GUI	Requests the needed parameters from the user and starts the conversion process.
GUIConvert	GUI shown when the conversion process has started. The GUI displays status and error messages.
Convert	Thread that steers the conversion process and also transforms the node screen coordinates to traffic map coordinates and determines the NodeActionType for each node.
TickInformation	Stores information about the nodes that are part of the tick that is currently processed.
Dataset	Writes the TickInformation and nodes contained by it to the AHS/AHV dataset.
Node	Class that contains information about a node.
DSData	Structure that is used to pass user input data through the program.

I.2 Implementation

The Convert IT application is a multi-threaded application that is not effected by the issue of synchronization as with the AHS and AHV programs. The converting of the coordinate information takes place in one thread that does not transfer data between threads except for text messages that are send to a read-only textbox.

The conversion of a dataset is described in the flow chart depicted in figure 35. The flowchart depicts the processing done after the user has set up the parameters required for the conversion process.

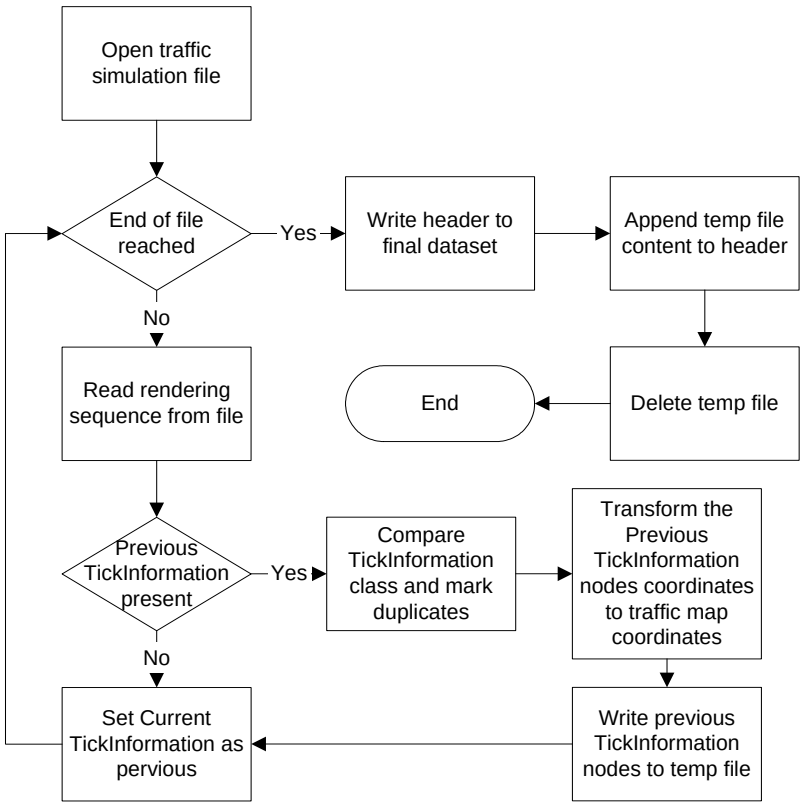

Figure 35: Flowchart of dataset conversion process

The coordinate transformation from traffic simulation generated screen coordinates to traffic map coordinates or normal coordinates is done with the function shown in figure 36. This function is the reverse of the function TransformToScreen function found in the traffic simulation program.

```
public static double TransformToNorm(double coord,
  double add, int pMin, int pMax, int pScreenSize,
  double pZoom, double pZoomCenter)
{
   if (pMax <= pMin)
          pMax = pMin + 100;

   return (((((coord + add) / pScreenSize) - pZoomCenter) /
          pZoom) + pZoomCenter) * ((pMax - pMin) + pMin);
}
```

Figure 36: TransformToNorm function

The normal coordinates calculated give the position of a node in traffic map coordinates. This enables the AHV rendering engine to draw nodes at the correct position on screen and the AHV and AHS location manager to calculate the exact distance between nodes, required for the nearest neighbour detection.

I.3 Dataset layout

The dataset layout used by the AHS and AHV program comprises a header, multiple ticks and node location information.

The dataset header consists out of eight parameters that are stored on the first line of the dataset; this line is prefixed with the letter 'H'. The first parameter of the header contains the name of the traffic simulation map file on which the simulation was run. The second parameter contains the number of ticks present in the file. The third parameter contains the maximum number of nodes active at any given time during the entire simulation.

```
H|test3.map|2506|36|120|120|0|0|80
T|0|1
0|0|99.7301472860233|39.2363011519124
T|1|3
0|1|99.359357297353|39.2363011519124
1|1|58.0224533937584|63.2164449818621
0|1|93.5503141415182|39.2363011519124
T|2|0
T|3|2
1|1|58.0224533937584|63.5219245210971
0|1|93.1795241528479|39.2363011519124
T|4|3
2|0|20.3934493768668|41.0691783873225
1|1|58.0224533937584|63.8274040603322
0|-1|92.8087341641776|39.2363011519124
```

Figure 37: Dataset example

The fourth parameter contains the largest value for the x-axis as found in the traffic map file on which the simulation was run. The fifth parameter contains the largest value for the y-axis as found in the traffic map file. The sixth parameter contains the smallest value for the x-axis as found in the traffic

map file. The seventh parameter contains the smallest value for the y-axis as found in the traffic map file. The eighth parameter contains the number of meters that one traffic map file unit represents.

The lines prefixed with the letter 'T' represent ticks. These ticks represent the time when the traffic simulation program started a new rendering sequence and updated the coordinates for the different nodes. A tick consists out of two parameters. The first parameter represents the current tick number, starting from zero. The second parameter represents the number of node mutations present in this tick.

The data stored in the dataset consists out of four parameters and is not prefixed. The first parameter represents the nodes 'unique' number. The second parameter represents the operation that needs to be carried out on the node. The third parameter represents the new coordinate for the x-axis. The fourth parameter represents the new coordinate for the y-axis.

There are three different types of operations that can be carried out on a node. The first operation is represented by the value zero and indicates that the node is new and needs to be created in memory before processing coordinate information. The second operation is represented by the value one and indicates that only the new coordinates need to be processed by the simulation. The third operation is represented by the value minus one and indicates that the node trajectory is completed and that it can be removed from memory.

APPENDIX II

USER MANUAL

This appendix introduces the basic steps needed to build a dataset for the AHS and AHV program by describing the functionality and use of the modified traffic simulation program and the Convert IT program. The AHS and AHV program GUI and options will also be discussed in detail.

II.1 Traffic Simulation

In order build a dataset with coordinate information a modified version of the traffic simulation program was developed. This program functions exactly like described in **[K02]** but requires two text files to be placed on the root directory of the c-drive of the Windows operating system. The files are named 'test.txt' and 'MapCoordTransform.txt'.

Once the above-mentioned files are present start the Traffic simulation program as normal. Select a city map, set the 'run' parameter to 1 and modify other settings at will, then run the simulation. While recording a simulation it is vital that you do **NOT** alter the zoom level or scrollbar positions, doing so will make the data useless.

II.2 Convert IT

In order to create a dataset with traffic information organized in an AHS/AHV readable format the Convert IT program is developed. Before data conversion can begin we first need to specify the dataset that holds the raw information. The steps needed to create this raw dataset are described in the previous paragraph.

By pressing the top 'Browse' button and selecting the 'test.txt' file this step is completed. Then we need to specify the traffic map used in the Traffic simulation program by pressing the lower 'Browse' button on the right side of the screen.

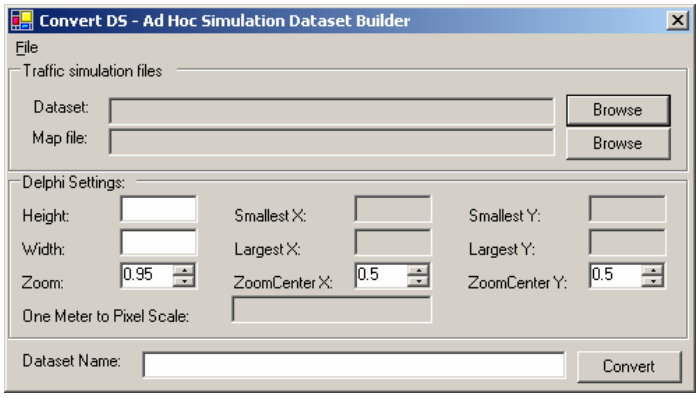

Figure 38: Convert IT program GUI

After registering the dataset and map file the Convert IT program will fill in most of the parameters required for the conversion. The only parameters the user needs to specify are the screen height and width of the rendering panel in the traffic simulation program, these value's can be found in the 'MapCoordTransform.txt' file. Finally we need to specify a dataset name and press the convert button.

After pressing the convert button a new window will open displaying the progress of the file conversion and any error's encountered. This conversion process can consume quite some time on slow PC's. As soon as the conversion process is complete the Convert IT application can be closed. The newly created dataset can be found in the working directory of the Convert IT program.

Copying the Dataset and traffic map files to the right directories of the AHS and AHV program completes the operation.

II.3 The options screen

The options screen used for AHS and AHV program have the same layout and settings therefore it will be discussed here separately of the AHS and AHV program. The options screen is accessible by pressing the 'F3' function button on the keyboard by navigating to the menu bar and choosing Simulation and the options. Once opened the screen as depicted below will appear.

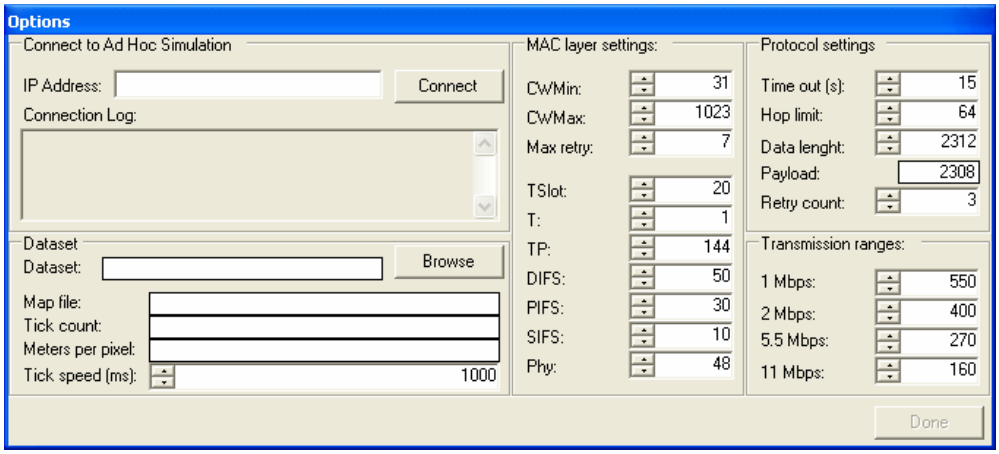

Figure 39: Options screen

The top left part of the screen sets up a TCP/IP network connection between the AHS and AHV program. By specifying an IP address and then pressing the connect button the connection is made. In case an error occurs during the setup process the application will appear to 'hang', do not close the application but wait until it returns to normal operation. This can take up to one minute. The bottom left part of the screen enables the user to load a dataset and set the read speed used by the AHS and AHV programs. To load a dataset press the 'Browse' button and navigate to the dataset using the then displayed window. The centre of screen holds the default parameters for the IEEE 802.11B MAC layer. These parameters can be modified by the user at will, but can lead to unexpected results. Please refer to appendix VIII for a complete description of the individual

parameters. The top right portion of the screen holds the ARA protocol influential parameters. The bottom right portion of the screen holds the transmission ranges specified for the different transmission speeds. Once the initial setup is done de 'Done' will be enabled. The simulation can now be run.

II.4 AHS

The AHS program runs the entire simulation of the PDA's, protocols and MAC layer's and the communication between them. Therefore the emphasis with the AHS program lies completely on the simulating part, resulting in a minimal interface.

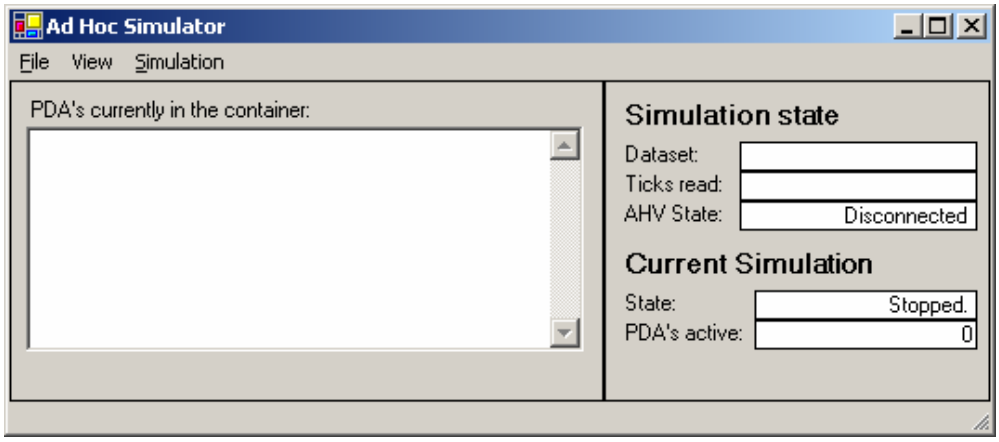

Figure 40: AHS program GUI

Figure 40 shows the AHS GUI as shown at program start. Once the simulation is setup using the options screen (see section three) the simulation can be run be navigating to Simulation and the start or by pressing the function key F5. The PDA's that are registered in the simulation environment are listed in the top left portion of the screen. By clicking on the name of a PDA its PDA GUI is displayed (see section six). The right portion of the screen gives information about the state of the simulation.

The log window is shown by selecting view and then log in the main menu or by pressing the function button 'F3'. The log windows will be attached to the bottom part of the screen displaying simulation progress messages and any errors that might occur during the simulation process.

II.5 AHV

The AHV program only visualizes the results like statistics and traffic flow as gathered during a simulation run with the AHS program. The AHV screen is divided into two sections. The right-hand section of the screen displays simulation progress statistics, Rendering options and graphs displaying up to date information about the number of active nodes, total number of packets send and total number of packets lost.

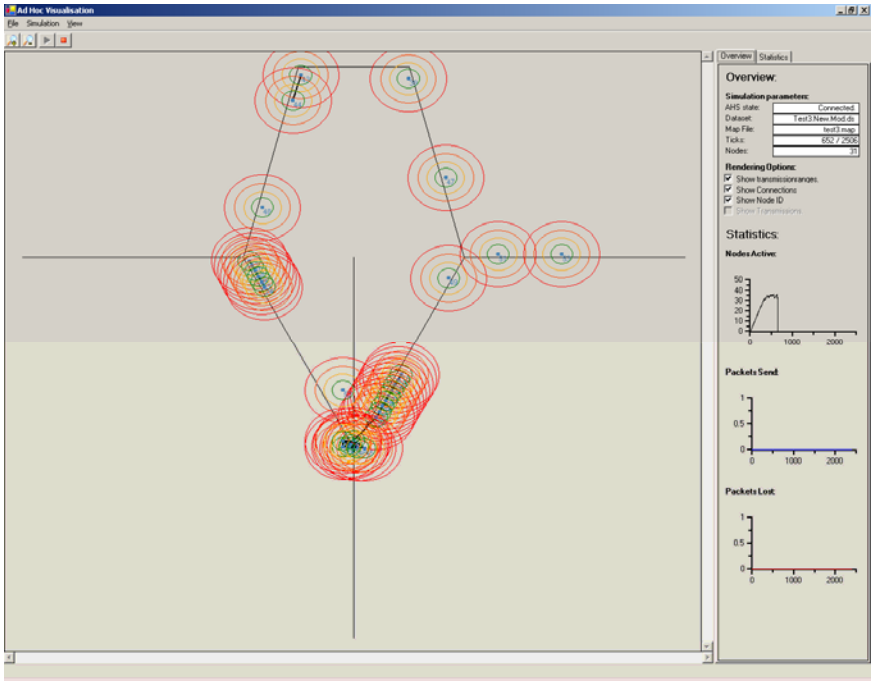

Figure 41: AHV program GUI

The main portion of the screen is taken up by the traffic network screen, which displays the cars moving along the city street network. Cars in the network are represented by a blue dot that follows a route along city street network. Wireless network transmission ranges, network connections and node numbers are rendered by default but can be deactivated using the rendering options on the right side of the screen.

The zoom in and out buttons on the toolbar, located in the top left portion of the screen allows the user to zoom in and out on the map. The vertical and horizontal scrollbars enable the user to focus on a specific portion of the map when zoomed in.

Once the simulation is setup using the options screen (see section three) the simulation can be started by pressing the green play button on the toolbar, the 'F5' button or by navigating to Simulation and then start.

II.6 PDA GUI

The PDA GUI displays the information gathered by the PDA on the moment the PDA GUI was accessed. The top left portion of the screen display the number and types of the packets send and received. The bottom left part of the screen show the nodes one hop neighbours. The right portion of the screen provides the user with the possibility to send files and text messages to other nodes in the network.

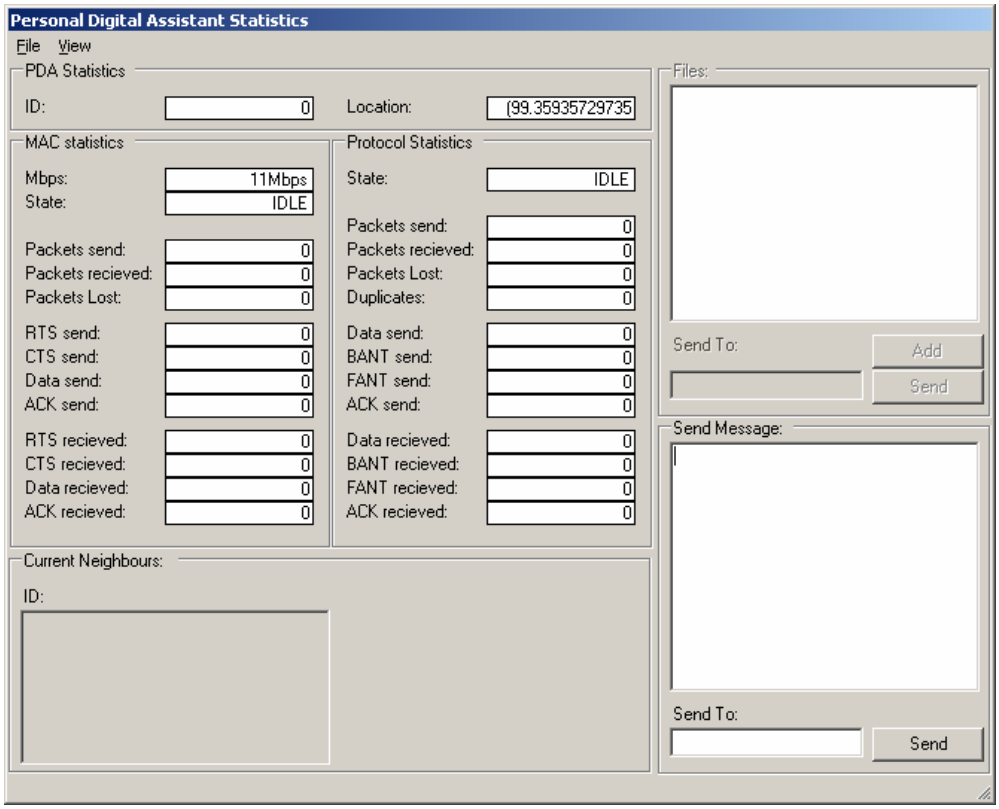

Figure 42: PDA GUI

On start up only the top left portion of the screen is shown, the current neighbours and transfer screen can be displayed by navigating to the view menu.

By navigating to the file menu and then pressing the 'Refresh' option will refresh the statistics to resemble the current up-to-date information. Navigating to file and then pressing the 'Exit' option will close the PDA GUI screen.

APPENDIX III

SECTOR BASED NEAREST NEIGHBOUR QUERIES

Wireless devices are limited in their communication capabilities by the maximum transmission range over which they can propagate their signal. To incorporate this feature into the simulation environment a nearest neighbour detection algorithm had to be implemented. This chapter discusses the algorithm implemented.

III.1 The algorithm

When searching for the nearest neighbours of a specified node the brute force method works well on small collections of nodes. The greater the numbers of nodes the more calculations have to be made in order to find the correct set of neighbours. The traffic map files used by the traffic simulation program together with the datasets used for the AHS/AHV program provide an elegant way to minimize the number of calculations.

Taking the maximum and minimum x-axis and y-axis coordinates out of the traffic-map file provides us with an imaginary rectangle around the entire city street network. By dividing this rectangle into squares all of equal size, the edge length of the square must always be greater than the diameter of the maximum transmission range for any given node and must also be a power of two. Each node is then made aware of its direct neighbours, these neighbours are stored in an array and are ordered as shown in figure 43.

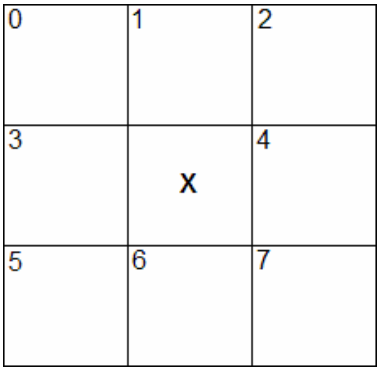

Figure 43: Neighbour numbering

The nodes are stored in the sector that encompasses the area in which the nodes coordinates fall. If a node now wants to retrieve its set of neighbours the following actions will be undertaken. Calculate the sector coordinates of the node by using the formula:

$$S(c) = N(c) \ \% \ Edge - Radius$$

Where S(c) is the sector coordinate, N(c) is the node coordinate, Edge is length of the sector edge and Radius is the maximum transmission range. These calculations must be done for the nodes x-axis and y-axis coordinates. The sign of the sector coordinates now specifies which sectors fall within the transmission range of the node as shown in table three, note that the sector in which to node lies is always searched.

Table 3: Search sectors based on sector coordinates.

x-axis	y-axis	Sectors
Negative	Negative	0, 1, 3
Positive	Negative	3, 5, 6
Negative	Positive	1, 2, 4
Positive	Positive	4, 6, 7

Now that the sectors that need to be searched are determined we perform the Euclidian distance metric on the member nodes of the sectors.

$$Dist = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

By adding all nodes with a distance smaller or equal to the maximum transmission range of the current node the set of nearest neighbours is retrieved.

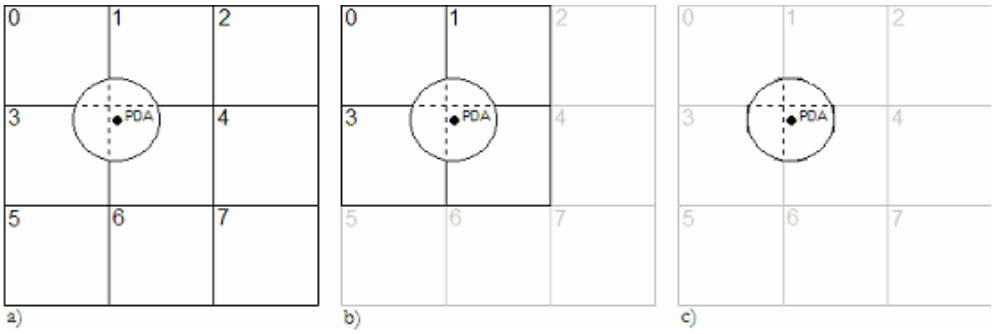

Figure 44: Different steps of neighbour calculation

APPENDIX IV

AHS

This appendix contains the class diagram of the AHS program, a data dictionary is also included explaining the purpose of each of the classes. The classes shown in the class diagram but not mentioned in the data dictionary are located in the common namespace and are described in chapter four section three.

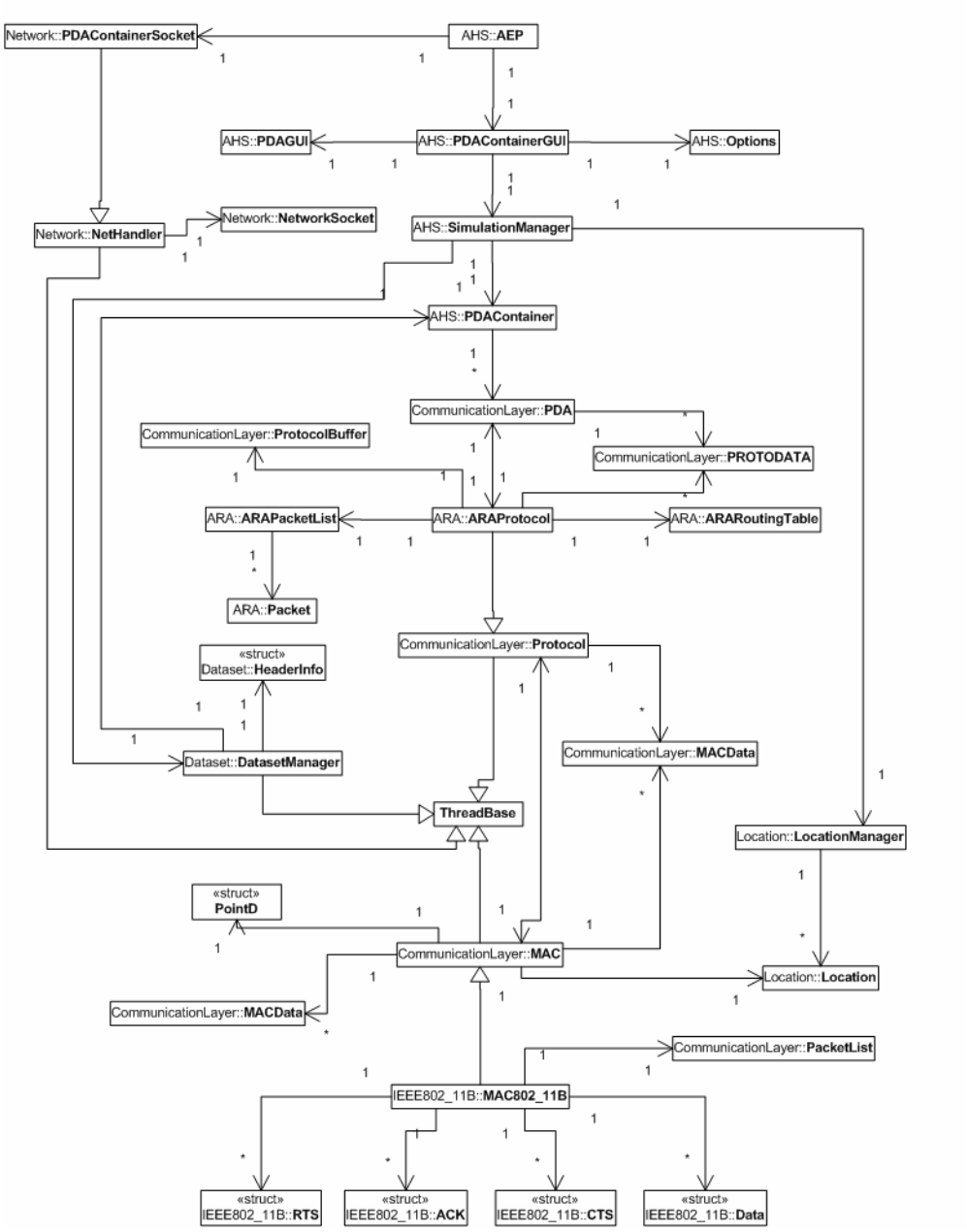

Figure 45: AHS class diagram

Table 4: AHS data dictionary

Class	Description
AEP	Application entry point. Contains the main function of the program and is responsible for the initial start up of the program.
ARAPacketlist	Filters out duplicate ants and data packets received by the MAC layer.
ARAProtocol	Protocol class that uses ants to establish routes between mobile nodes in order to transmit data between them.
ARARoutingTable	Routing table that stores routes between nodes.
Datasetmanager	Class that loads node coordinate information from a dataset at specified intervals.
Location	Stores all nodes that lie within its coordinates, also locates the neighbours of a node by querying itself and the neighbour locations.
Locationmanager	Initializes, contains and provides access to the locations.
MAC	Extendable model of a media access layer.
MAC802.11B	Basic implementation of a media access layer based on the IEEE 802.11B specification.
MACData	Class used to exchange data between MAC and Protocol. Stores data transmission details like destination and next hop.
Options	Options screen, allows the user to alter simulation parameters and dataset to use.
PacketList	Filters out duplicate packets.
PDA	Class that implements a minimal PDA.
PDAContainer	Class stores the PDA's that loaded and provides a way to access them.
PDAContainerGUI	GUI that display's the current state of

	the AHS environment and the PDA's registered in the PDAContainer.
PDAContainerSocket	TCP/IP Socket wrapper class, this class extends functionality provided by the Nethandler class.
PDAGUI	GUI that display's the statistics of the PDA in question. Also provides a means to send messages.
PointD	2D Vector representing the coordinates of a node in the double (precision of 15 ~ 16 digits) format.
Protocol	Extendable model of a Protocol
ProtocolBuffer	Packet storage facility stores received packets in a stream until data transmission is complete.
Protodata	Class that stores information supplied by the PDA that needs to be transmitted to another node.
Simulationmanager	Initializes, starts and stops the simulation.

APPENDIX V

AHV

This appendix contains the class diagram of the AHV program, a data dictionary is also included explaining the purpose of each of the classes. The classes shown in the class diagram but not mentioned in the data dictionary are located in the common namespace and are described in chapter four section three.

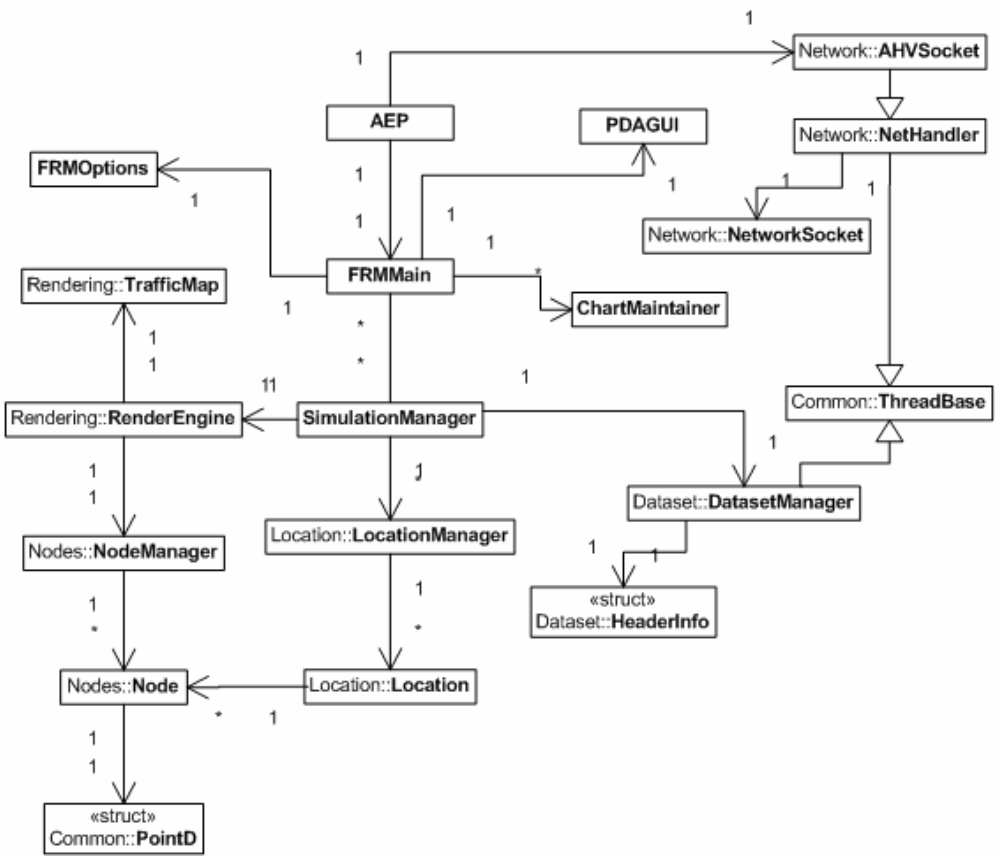

Figure 46: AHV class diagram

Table 5: AHV data dictionary

Lass	Description
AEP	Application entry point. Contains the main function of the program and is responsible for the initial start up of the program.
AHVSocket	TCP/IP network socket that enables the AHV program to communicate with the AHS program.
ChartMaintainer	Provides easy access functions for updating and mutating chart present in the AHV program

Datasetmanager	Class that loads node coordinate information from a dataset at specified intervals.
FRMMain	The main GUI of program display's statistics and the city street network with moving nodes.
Location	Stores all nodes that lie within its coordinates, also locates the neighbours of a node by querying itself and the neighbour locations.
Locationmanager	Initializes, contains and provides access to the locations.
Node	Class representing a node in the simulation environment.
Nodemanager	Class that maintains a array of nodes and provides functionality for adding, removing and updating nodes.
PDAGUI	GUI that display's the statistics of the PDA in question. Also provides a means to send messages.
PointD	2D Vector representing the coordinates of a node in the double (precision of 15 ~ 16 digits) format.
RenderEngine	Memory buffered rendering engine that renders the traffic network and nodes to screen.
Simulationmanager	Initializes, starts and stops the simulation.
Trafficmap	Memory buffered class that renders the city street network to memory.

APPENDIX VI

MAC LAYER FLOWCHARTS

The three flow charts depicted on the following pages show in detail the program flow each different type of packet has to travel when send to or received from the MAC layer.

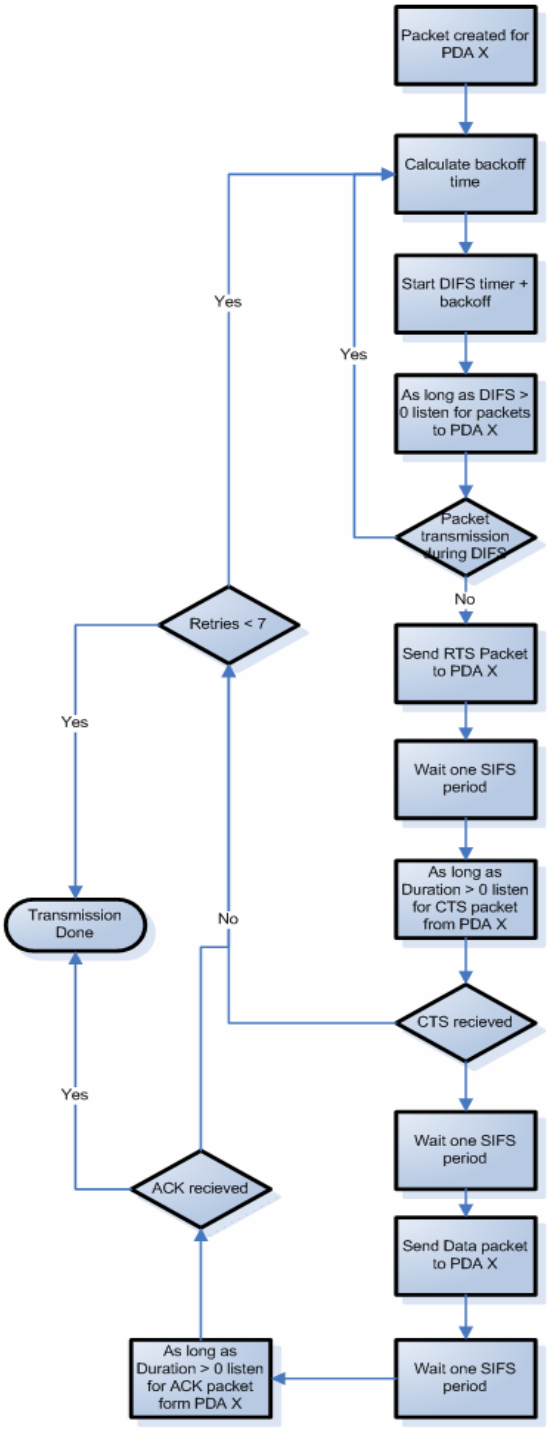

Figure 47: Data transmission sequence

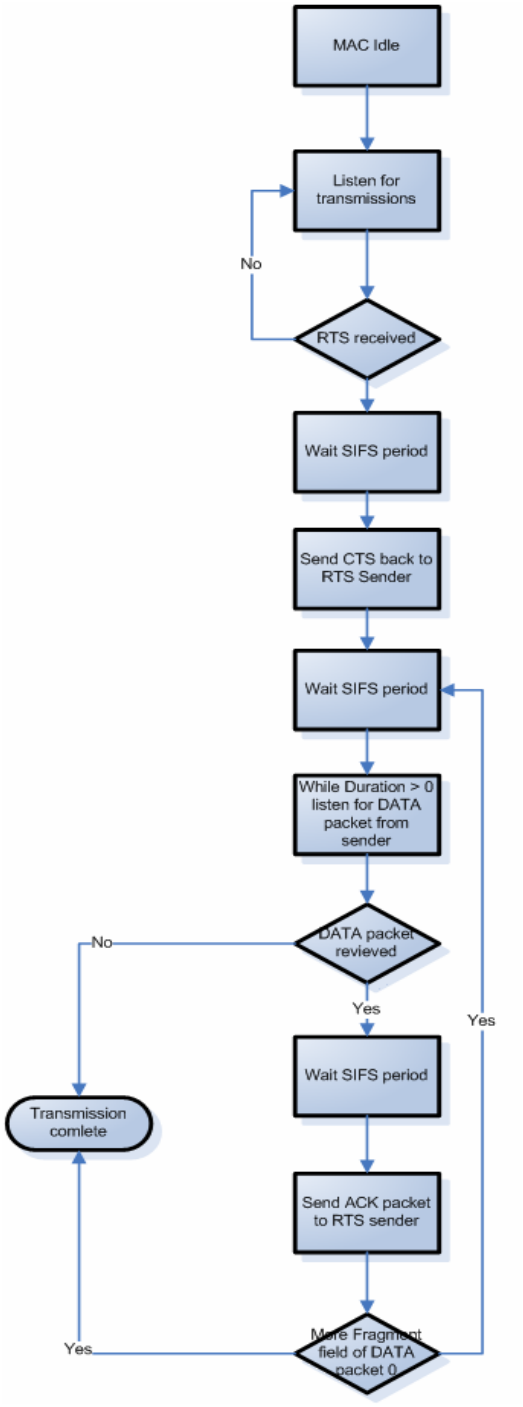

Figure 48: Data reception sequence

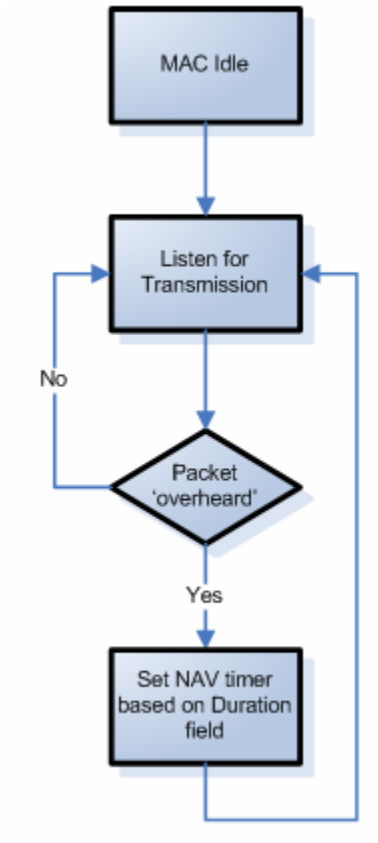

Figure 49: Packet overhearing; Idle state

APPENDIX VII

ARAPROTOCOL FLOWCHARTS

The four flow charts depicted on the following pages show in detail the program flow each different type of packet has to travel when send to or received from the MAC layer.

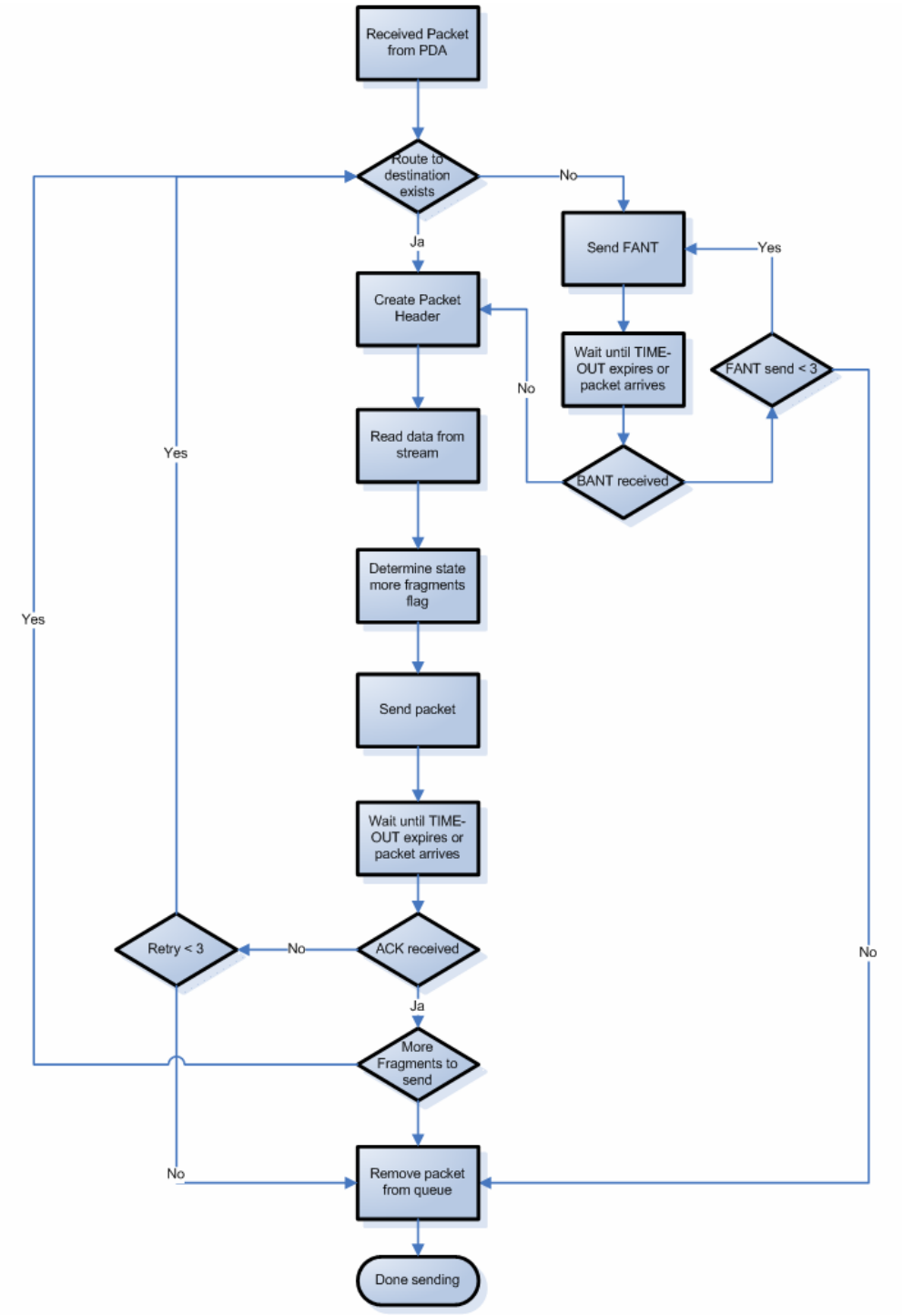

Figure 50: Outgoing data packet

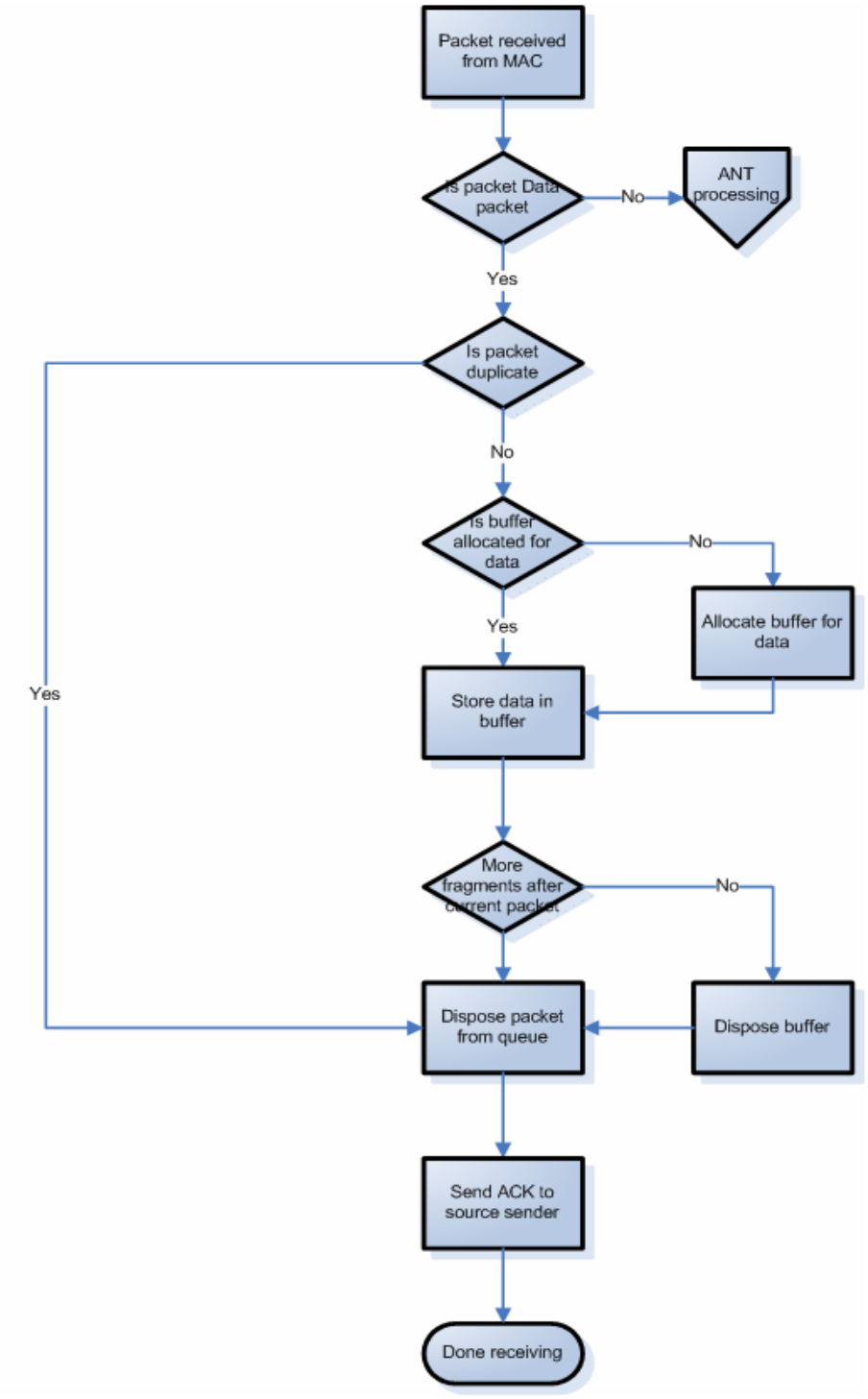

Figure 51: Incoming data packet

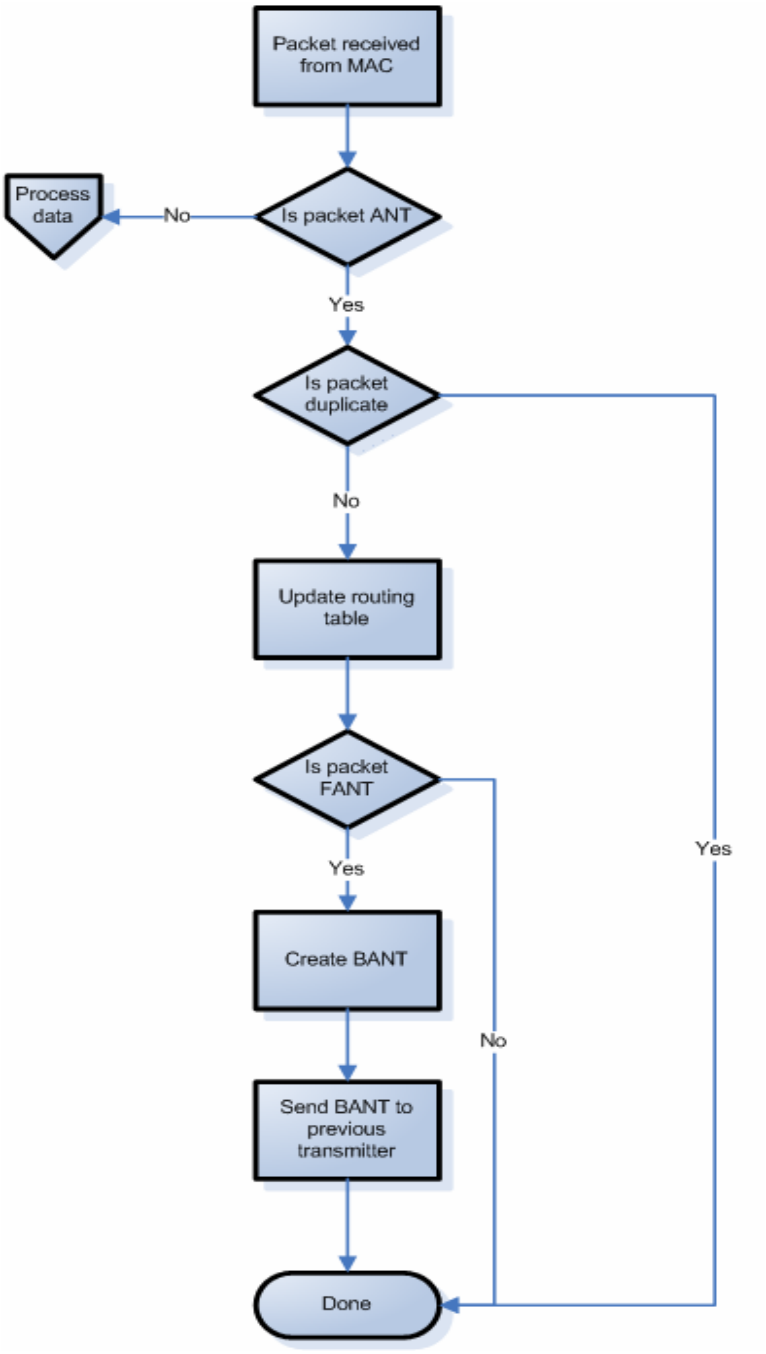

Figure 52: Incoming ant data packet.

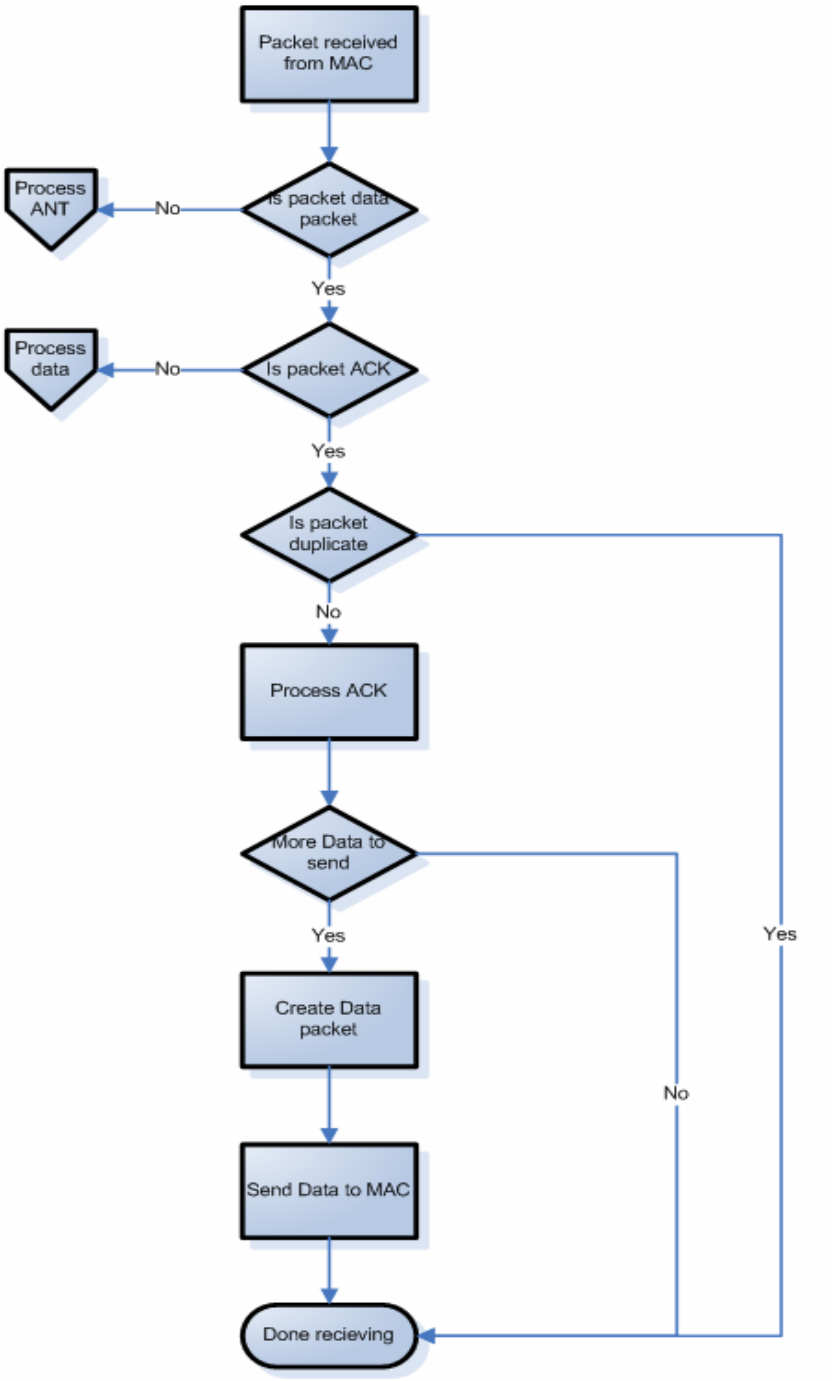

Figure 53: Incoming ACK packet

APPENDIX VIII

IEEE 802.11B

This appendix lists the default values as defined by the IEEE 802.11B specification for the various parameters that influence the MAC layer. Table six specifies the default parameters set for the transmission timing calculations.

Parameter	Value	Description
Tslot	20 µs	Slot time
τ	1 µs	Propagation delay
Tp	144 µs	Transmission time of the physical preamble.
Tdifs	50 µs	DIFS time
Tsifs	10 µs	SIFS time
Tpifs	30 µs	PIFS time
CWmin	31	Minimum back off window size.
CWmax	1023	Maximum back off window size.
Tphy	48 µs	Transmission time of physical header

Table 6: MAC layer default parameters

Table seven lists the transmission speeds in relation to transmission range. Note that the transmission ranges in table seven are only possible in an ideal environment, signal interference will cause these transmission ranges to decrease drastically in a real life situation.

Table 7: MAC layer ideal transmission ranges

Transmission speed	Communication range
1 Mbps	550 meter (601.49 yards)
2 Mbps	400 meter (437.45 yards)
5.5 Mbps	270 meter (295.28 yards)
11 Mbps	160 meter (174.98 yards)

Table eight Lists the type and sub type numbers associated with the different packets transmittable in the implemented version of the MAC layer.

Table 8: Packet type and subtype ids

Type	Description	Subtype	Description
01	Control	1011	RTS
01	Control	1100	CTS
01	Control	1101	ACK
10	Data	0000	Data

APPENDIX IX

MAC TEST RESULTS

```
Started
1: OutgoingPacket
1: SendRTS
1: Timer set 670
1: SendToMac
1: IDLE
2: ProcessIncommingPacket
2: ProcessRTS
2: NAV set: 670
2: Timer set 670
2: IDLE
1 Time Elapsed
1: OutgoingPacket
1: SendRTS
1: Timer set 733
1: SendToMac
1: IDLE
2: ProcessIncommingPacket
2: ProcessRTS
2: NAV set: 733
2: Timer set 733
2: IDLE
3: ProcessIncommingPacket
3: ProcessRTS
3: NAV set: 733
3: Timer set 733
3: IDLE
2 Time Elapsed
1 Time Elapsed
1: OutgoingPacket
1: SendRTS
1: Timer set 950
1: SendToMac
3: ProcessIncommingPacket
3: ProcessRTS
3: NAV set: 950
3: Timer set 950
1: IDLE
3: IDLE
2: ProcessIncommingPacket
2: ProcessRTS
2: NAV set: 950
2: Timer set 950
2: IDLE
1 Time Elapsed
1: OutgoingPacket
1: SendRTS
1: Timer set 1294
1: SendToMac
```

```
3: ProcessIncommingPacket
3: ProcessRTS
3: NAV set: 1294
3: Timer set 1294
4: ProcessIncommingPacket
4: ProcessRTS
4: SendCTS
4: Timer set 932
1: IDLE
2 Time Elapsed
3: IDLE
2: ProcessIncommingPacket
2: ProcessRTS
2: NAV set: 1294
2: Timer set 1294
2: IDLE
4: SendToMac
4: IDLE
3: ProcessIncommingPacket
3: processCTS
3: NAV SET CTS
3: Timer set 932
1: ProcessIncommingPacket
1: processCTS
1: SendData
1: Timer set 314
2: ProcessIncommingPacket
2: processCTS
2: NAV SET CTS
2: Timer set 932
3: IDLE
2: IDLE
1: SendDataToMac 0
4: ProcessIncommingPacket
4: ProcessData
//----------------------------------------
1: IDLE
// 5/5/2004 3:16:50 PM - Macdata Received
// Signal   : -1
// Address1 : 1
// Address2 : 4
// Address34: 1
// Data     : TEST

TEST
// Sequence : 1
// Fragment : 0
// More     : False
// MF       : True
//----------------------------------------
4: SendAck
4: SendToMac
2: ProcessIncommingPacket
2: ProcessACK
2: IDLE
3: ProcessIncommingPacket
3: ProcessACK
3: IDLE
4: IDLE
1: ProcessIncommingPacket
1: ProcessACK
1: IDLE
Stopped
```

APPENDIX X

ARAPROTOCOL TEST RESULTS

```
Started
BANT_WAIT Timer Set
Protocol: 0: Ant send. BANT Time-out Set
1: Received Packet
Protocol: 1 Received: 3 | 0 |FANT
Protocol: 1: Processing ANT 1: 0
0: Received Packet
2: Received Packet
Protocol: 0 Received: 3 | 0 |FANT
Protocol: 0: Processing ANT 0: 0
Protocol: 0: Duplicate dropped.
Protocol: 2 Received: 3 | 0 |FANT
Protocol: 2: Processing ANT 2: 0
1: Received Packet
3: Received Packet
Protocol: 1 Received: 3 | 0 |FANT
Protocol: 1: Processing ANT 1: 0
Protocol: 1: Duplicate dropped.
Protocol: 3 Received: 3 | 0 |FANT
Protocol: 3: Processing ANT 3: 0
Protocol: 3: Sending BANT
2: Received Packet
Protocol: 2 Received: 0 | 3 |BANT
Protocol: 2: Processing ANT 2: 3
1: Received Packet
3: Received Packet
Protocol: 1 Received: 0 | 3 |BANT
Protocol: 1: Processing ANT 1: 3
Protocol: 3 Received: 0 | 3 |BANT
Protocol: 3: Processing ANT 3: 3
Protocol: 3: Duplicate dropped.
0: Received Packet
Protocol: 0 Received: 0 | 3 |BANT
Protocol: 0: Processing ANT 0: 3
Protocol: 0: route found: 1
Protocol: 0ACK_WAIT Timer Set
2: Received Packet
Protocol: 2 Received: 0 | 3 |BANT
Protocol: 2: Processing ANT 2: 3
Protocol: 2: Duplicate dropped.
Reading position: 2295 - Fragment 1 done: 2312
Sending fragment to: 3, via: 1
1: Received Packet
Protocol: 1 Received: 3 | 0 |DATA
2: Received Packet
Protocol: 2 Received: 3 | 0 |DATA
Tick
3: Received Packet
Protocol: 3 Received: 3 | 0 |DATA
```

```
Protocol: 3: ACK send
2: Received Packet
Protocol: 2 Received: 0 | 0 |ACK
1: Received Packet
Protocol: 1 Received: 0 | 0 |ACK
0: Received Packet
Protocol: 0 Received: 0 | 0 |ACK
Protocol: 0: route found: 1
Protocol: 0ACK_WAIT Timer Set
Reading position: 4603 - Fragment 2 done: 2312
Sending fragment to: 3, via: 1
1: Received Packet
Protocol: 1 Received: 3 | 0 |DATA
2: Received Packet
Protocol: 2 Received: 3 | 0 |DATA
Tick
3: Received Packet
Protocol: 3 Received: 3 | 0 |DATA
Protocol: 3: ACK send
2: Received Packet
Protocol: 2 Received: 0 | 0 |ACK
1: Received Packet
Protocol: 1 Received: 0 | 0 |ACK
0: Received Packet
Protocol: 0 Received: 0 | 0 |ACK
Protocol: 0: route found: 1
Protocol: 0ACK_WAIT Timer Set
Reading position: 6911 - Fragment 3 done: 2312
Sending fragment to: 3, via: 1
1: Received Packet
Protocol: 1 Received: 3 | 0 |DATA
2: Received Packet
Protocol: 2 Received: 3 | 0 |DATA
Tick
3: Received Packet
Protocol: 3 Received: 3 | 0 |DATA
Protocol: 3: ACK send
2: Received Packet
Protocol: 2 Received: 0 | 0 |ACK
1: Received Packet
Protocol: 1 Received: 0 | 0 |ACK
0: Received Packet
Protocol: 0 Received: 0 | 0 |ACK
Protocol: 0: route found: 1
Protocol: 0ACK_WAIT Timer Set
Reading position: 9219 - Fragment 4 done: 2312
Sending fragment to: 3, via: 1
1: Received Packet
Protocol: 1 Received: 3 | 0 |DATA
Tick
2: Received Packet
Protocol: 2 Received: 3 | 0 |DATA
3: Received Packet
Protocol: 3 Received: 3 | 0 |DATA
Protocol: 3: ACK send
2: Received Packet
Protocol: 2 Received: 0 | 0 |ACK
1: Received Packet
Protocol: 1 Received: 0 | 0 |ACK
0: Received Packet
Protocol: 0 Received: 0 | 0 |ACK

......
```

```
Reading position: 40960 - Fragment 18 done: 1741
Sending fragment to: 3, via: 1
1: Received Packet
Protocol: 1 Received: 3 | 0 |DATA
Tick
2: Received Packet
Protocol: 2 Received: 3 | 0 |DATA
3: Received Packet
Protocol: 3 Received: 3 | 0 |DATA
Protocol: 3: ACK send
2: Received Packet
Protocol: 2 Received: 0 | 0 |ACK
1: Received Packet
Protocol: 1 Received: 0 | 0 |ACK
0: Received Packet
Protocol: 0 Received: 0 | 0 |ACK
Stopped
```