Faculdade de Engenharia da Universidade do Porto

Mobile Router

Control Plane

Daniel Guedes da Costa Neves Carrapa

Project Report developed for the
Integrated Masters in Computer and Electrical Engineering
Major in Telecommunications

Supervisor: Prof. Dr. Manuel Alberto Pereira Ricardo
Co-Supervisor: Filipe Miguel Monteiro da Silva e Sousa
Co-Supervisor: Gustavo João Alves Marques Carneiro

June 2009

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES 2008/2009

A Dissertação intitulada

"MOBILE ROUTER"

foi aprovada em provas realizadas em 22/Julho/2009

o júri

Presidente Professor Doutor José António Ruela Simões Fernandes
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

Professor Doutor Rui Luis Andrade Aguiar
Professor Associado do Departamento de Engenharia Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Professor Doutor Manuel Alberto Pereira Ricardo
Professor Associado do Departamento de Engenharia Electrotécnica e de Computadores da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projecto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extractos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são correctamente citados.

Autor - **DANIEL GUEDES DA COSTA NEVES CARRAPA**

Faculdade de Engenharia da Universidade do Porto

Resumo

Grandes redes emalhadas que providenciam acessos de banda larga à internet e meios para comunicação de grande débito a terminais em áreas metropolitanas precisam de uma arquitectura robusta e expansível usando nós de encaminhamento eficientes entre os terminais. Os protocolos de encaminhamento actuais são incapazes de lidar com redes de milhares de terminais eficientemente e com fiabilidade. Esta dissertação apresenta um projecto de implementação do plano de controlo de um desses nós usando um novo protocolo de encaminhamento para realizar uma rede onde se procura dar acessos de banda larga a milhares de clientes de um sistema de transportes públicos de maneira segura e fiável.

A implementação foi feita através da emulação da versão de simulação do protocolo e análise do seu comportamento numa *testbed*. Os resultados mostram a adaptabilidade do protocolo a situações de mobilidade e mudanças na topologia da rede.

Este documento apresenta a tese de dissertação do Mestrado Integrado de Engenharia Electrotécnica e de Computadores. Neste documento serão apresentadas soluções existentes para os ambientes actuais de encaminhamento móvel e o funcionamento e resultados da implementação de uma nova solução.

Abstract

Large mesh networks to provide broadband access to the internet and high data rate for other means of communication to metropolitan areas terminals need a robust and scalable architecture using efficient routing nodes between terminals. Current routing protocols are incapable of dealing with networks of thousands of terminals efficiently and reliably. This dissertation presents a project to implement the control plane in one of such nodes using a new routing protocol to achieve a network that provides broadband access to thousands of clients in a public transportation system in a robust, safe and reliable way.

The implementation was achieved via emulation of a simulated version of the protocol and analysis of its behavior in a testbed scenario. The results show the adaptability of the protocol to mobility and changes in the network topology.

This document presents the dissertation thesis of the Integrated Master Program in Electrical and Computer Engineering. In this document existing solutions for mobile routing scenarios will be presented and also the working and results of a new solution.

Index

Index of figures

Index of tables

Abbreviations

List of abbreviations (by alphabetic order)

AF	Address Family
ANSN	Advertised Neighbor Sequence Number
AODV	Ad Hoc On Demand Vector Routing
ARP	Address Resolution Protocol
CPU	Central processing Unit
CR-LDP	Constraint-based Routing Label Distribution Protocol
DHCP	Dynamic Host Configuration Protocol
DRB	Designated Rbridge
DYMO	Dynamic MANET On-Demand
FEC	Forwarding Equivalent Class
GB	Giga Bytes
GNU	GNU is Not Unix
GPL	General Public License
HC	Hop Count
Htime	Hold Time
IC	IP Control
ID	Identifier
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
Inc	Incorporated
IS-IS	Intermediate System to Intermediate System
IP	Internet Protocol
LC	Logical Clock
LDP	Label Distribution Protocol
LER	Label Edge Router
LSP	Link State PDU
LSR	Label Switching Router
MAC	Media Access Control

MANET	Mobile Ad Hoc Network
MC	MAC Control
MPLS	Multiprotocol Label Switching
MPR	Multipoint Relay
MS	Message Size
Msg	Message
MSN	Message Sequence Number
NetDevice	Network Device
NID	Neighbor Identifier
NS3	Network Simulator 3
NS2	Network Simulator 2
OLSR	Optimized Link State Routing Protocol
Orig	Originator
OSI	Open Systems Interconnection
P2P	Peer to Peer
PDU	Protocol Data Unit
PDA	Personal Digital Assistant
QoS	Quality of Service
RAM	Random Access Memory
Rbridge	RoutingBridge
REBlock	Route Element Block
RERR	Route Error
RREP	Route Reply
RREQ	Route request
RSVP	Resource Reservation Protocol
TC	Topology Control
TRILL	Transparent Interconnection of Lots of Links
TTL	Time to Live
UDP	User Datagram Protocol
VLAN	Virtual Local Area Network
VPN	Virtual Private Networks
Vtime	Validity Time
WiMetroNet	Wireless Metropolitan Network
WMAN	Wireless Metropolitan Area Networks
WMRP	Wireless Metropolitan Routing Protocol
WNMT	Wireless Network for Metropolitan Transports

1. Introduction

1.1. Background

Currently mobile devices and mobile hardware using heterogeneous wireless technologies are increasing very rapidly. Cell-phones, PDAs, laptops and many other devices are growing in popularity inside big cities where the use of public transportation is a necessity due to high density of population. New problems arise when dealing with metropolitan networks as large as the ones in large cities to provide access to such devices. The search for a protocol to adapt to tens of thousands of mobile devices inside a metropolitan area with vehicular access is one of the most significant aspects of communication inside developed cities. The possibility of providing such access by a public transportation system to its vehicles, stations and passengers is trying to be achieved through the WiMetroNet, a Wireless Metropolitan Network. [1]

The WiMetroNet is a network designed especially for transportation systems. As the system has vehicles and stops, if a node is to be associated to each vehicle or stop that node is the WiMetroNet's interface with public terminals. These nodes have to be especially designed to be able to deal with different technologies. The chosen node for this network was the Rbridge that allows for traffic for OSI Model layer 2 and layer 3. Rbridges connected to each other maintain the connections from terminals to the network and consist of the core of the WiMetroNet. Inside the core an efficient routing protocol must be able to deal with thousands of terminals and up to two thousand Rbridges maintaining active routes. The protocol chosen for this network was Wireless Metropolitan Routing Protocol (WMRP). This protocol has shown better results in simulation than current routing protocols. To deal with the communications inside the network with data a data plane was developed based in MPLS, a network technology that provides fast and efficient data transmission.

1.2. Work Objectives

The WMRP is currently only implemented in a simulated environment. The desired solution is to know if the WMRP can successfully be implemented in real machines to support testbed scenarios correctly. This means that the solution would evaluate the control plane's reliability under scrutiny. Because the final solution includes a control plane and a data plane the solution developed here would only solve part of the problem.

1.3. Relevant Contributions

The chosen way to approach this evaluation was emulating by using machines running the simulator with only one Rbridge inside each one, and the interfaces of the underlying machine would provide for interfaces to physical networks.

1.4. Structure

This report is structured in five chapters. It begins by an introductory chapter that reveals the purpose of this project and its results. The second chapter will evaluate existing solutions for routing protocols and their problems adapting to the desired network. It will also briefly address the chosen routing protocol to implement, as well as the tool used for implementing the protocol, in order to understand where the solution had to be adapted to. The third chapter will present the work developed, the initial problem to be solved, how the solution was devised and a schematic that presents how the solution works. In the fourth chapter, tests to the implementation are described and their results are analysed. The final chapter provides conclusions about the work developed its interest for the solution of the described problem and future works that can in turn provide a better solution.

2. State of the Art

In this chapter we will review two important parts of the problem, the routing protocols that can be used in the WiMetroNet and the simulator for which it has been developed.

2.1. Routing in mobile and wireless networks

In this section we review existing protocols and network technologies that are related to the problem at hand. In the end the adopted solution will be presented, the Wireless Metropolitan Routing Protocol. After that a few comparisons between them and analysis will show what some of their problems are and which aspects of them could be chosen to be used in the current solution for the desired network.

2.1.1. Multi Protocol Label Switching (MPLS)

We will first consider MPLS as it is one of the current network technologies available with the less overhead when bridging frames.

MPLS is a network technology considered to act at an OSI Model layer connoted with the term “layer 2.5” in that it provides union between different layer 3 protocols and layer 2 protocols. It basically adds a header to the packet containing an identifying label. MPLS labels can be stacked to be able to tunnel through different networks.

MPLS is considered in [2] a solution to scalability and enables significant flexibility in routing. Due to its hierarchical possibilities it can easily enable high quality end-to-end service features that are required in different applications like virtual private networks. Traffic engineering made these benefits of MPLS networking possible. And nowadays the Constraint-based Routing Label Distribution Protocol (CR-LDP) and the ReSource Reservation Protocol (RSVP) are the signaling algorithms used for traffic engineering.

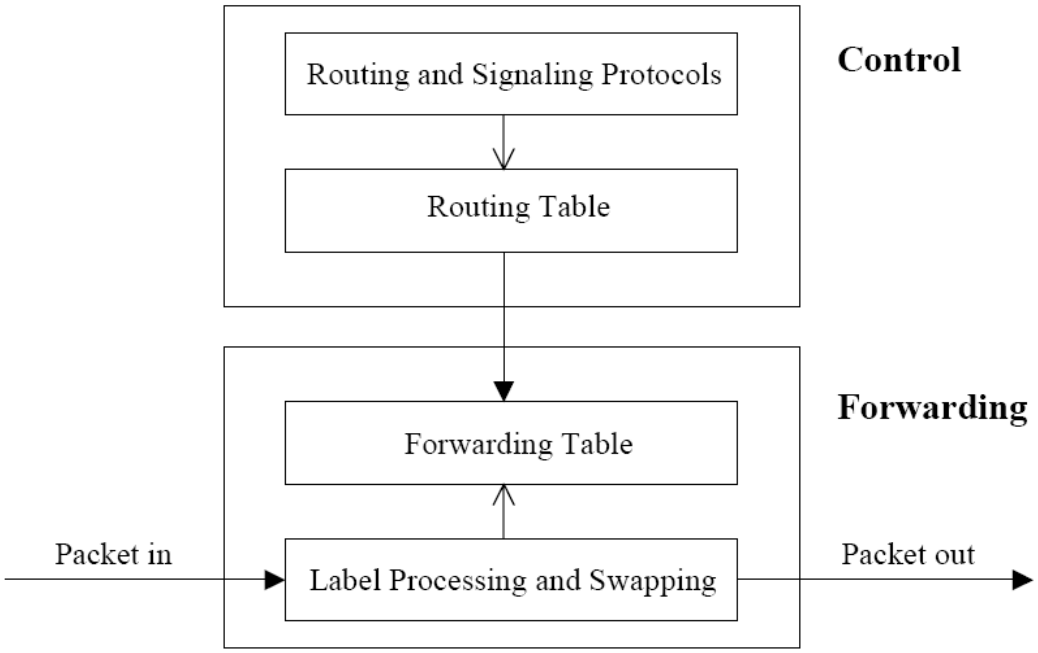


Figure 1 – MPLS Mapping between routing and forwarding tables [3]

In MPLS, the control and forwarding components are connected but function independently. The label is swapped based on the forwarding table that may be different from the routing table, depending on the forwarding table construction. As MPLS was created at a time in which label switching was faster than a routing table lookup, it was a faster way for packets to transverse large mesh networks with a lot of intermediary nodes; at that time routing table lookup was made in the CPU while label lookup could be made in the switch fabric. Because label lookup can be made at a lower level in the OSI model it is still faster than a routing table lookup.

The label in MPLS is the identifier of a forwarding equivalence class (FEC), an identifier of packets that share common characteristics, usually meaning they should be assigned the same label. These classes will correspond to a particular Label Switched Path (LSP).

MPLS acts within areas in the network, between an ingress node and egress node. Ingress nodes have identifying labels that are read when a packet reaches an intermediary node; it is then forwarded to the corresponding egress node with the identified label, where the MPLS header is decapsulated. The entry and exit nodes are called the Label Edge Routers (LER) which introduce a new MPLS header with a new label in the ingress node and remove it in the egress (exit) node.

The intermediary nodes, which only perform label switching, are called Label Switching Routers (LSR, in accordance with their functionality). The path corresponding to a

particular ingress node, intermediary nodes and egress node that share a label is called a LSP which doesn't mean that path cannot be used by other labels.

MPLS will apply when a packet arrives at an ingress node, a LER, in which a new label will be assigned to the packet according to its FEC and will forward it to the corresponding next LSR in the LSP. Each node in the LSP will examine the topmost label and according to the contents on that label the node will perform a different action: a swap, in which a label is switched with a new label and forwarded to the corresponding LSP; a push, in which a new label is introduced in the packet, creating a new MPLS layer which permits hierarchical ways of routing the packets permitting tunnels through different domains; and a pop, in which the topmost label is removed possibly ending the MPLS "tunnel" (usually attributed to the egress node) or a lower layer of MPLS. The default label assignment is the IETF standard LDP which sets and maintains the LSPs. Ensuring that adjacent nodes share a common FEC to label binding and allowing the creation of LSPs.[3] The forwarding table is then constructed based in the label configuration and the routing table (since the label configuration is based in the routing table).There are different reasons for the binding of certain nodes with certain labels usually based in routing parameters there are other possible reasons, like data traffic.

The following picture shows us the position of the MPLS header in a packet, and its encapsulation:

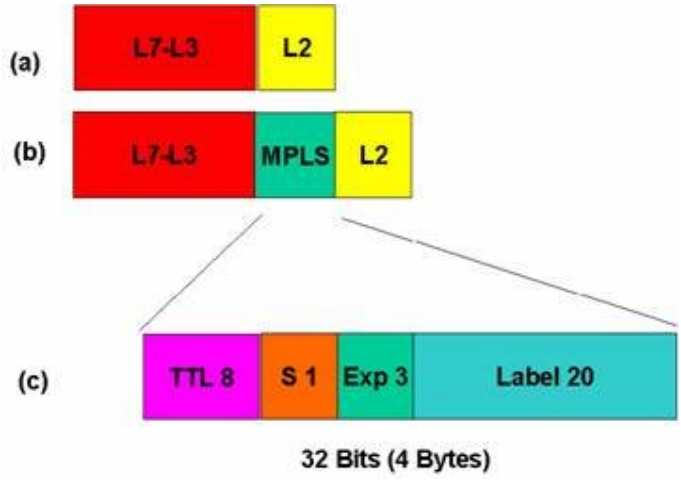


Figure 2 - MPLS header and encapsulation in a packet header [4]

Therefore, MPLS can deal with structured addresses but doesn't support automatically configured addresses; the label doesn't have to be connected to a layer 3 address however. It can control flooding as it has a TTL field. The scope of addresses is link based. And it can function flexibly under different layer 2 protocols.

The use of 20 bits for labels and the method of functioning mean it can provide communication for thousands of terminals. Its topology isn't structured and the fast label switching methods allow for improved performance. It is capable of supporting traffic

engineered paths with differentiation in service thanks to the label stacking, which provides a greater control of QoS. However, it does not support mobility under the current versions at the control plane because of the use of RSVP that needs some routing protocol to determine routes.

To summarize, MPLS can solve the problem of scalability, which means fulfilling the requisite to support more than ten thousand terminals inside a metropolitan area; the problem of flexibility, which means fulfilling the requisite of supporting different wireless technologies; the problem of flooding, which means fulfilling the requisite of dealing and not spreading broadcast traffic; although it cannot deal with the requisite of mobility and the dynamic configuration of terminals and nodes it is also important to note that the configuration protocol of MPLS needs routing protocols in each node participating in the network. Therefore the data plane part of MPLS was considered a partial solution for the problem and due to the easiness of separating data and control planes in MPLS it would only be necessary to build a control plane.

2.1.2. OLSR

We will now consider a routing protocol that can solve some problems regarding the routing problems in WiMetroNet.

OLSR is a proactive link-state routing protocol intended for mobile ad hoc networks although it can be used in different wireless ad hoc networks. It works as a protocol in which special nodes, MPRs (Multi Point Relays) are the only nodes that forward link state information.

It was made to decrease the amount of messages transmitted during a flooding process to achieve convergence in the network due to the native regular exchange of topology. MPRs are chosen by other nodes in its symmetric 1-hop neighborhood to perform these tasks and they declare the links to their MPR selectors. MPRs send control messages regularly to declare they are the chosen MPRs and they give the ability to reach their selectors to the rest of the network. OLSR works independently of the link-layer protocol, therefore it can work under different underlying link-layers. The use of MPRs, as the only nodes retransmitting control messages, minimizes flooding and overhead in the convergence process. It only requires partial link state to be flooded to provide the shortest path route. The protocol amount of messages transferred can be decreased by reducing the time interval between transmissions of control messages in order to decrease overhead, which results in a higher convergence time. This protocol is completely distributed and requires no central entity to control the topology or to monitor the network status. It also doesn't need reliable delivery of messages due to frequent exchange of control messages (mentioned above that this time may be altered), nor sequenced messaging because it possesses a sequence number

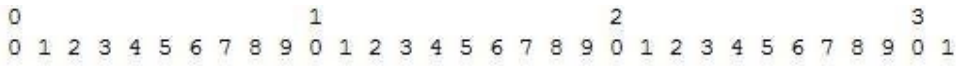
that enables nodes to identify and compare messages arriving. The protocol also doesn't interact with data sent in packets; it only works in a control plane.

In [5] we see that the algorithm of the selection of MPRs is close to the optimal solution. The number of isolated points is very important in that it may lead to a lack of robustness. Each node chooses a MPR set between its 1-hop neighbors in a way that covers all symmetric 2-hop nodes. From that subset the choice is made of the set with less MPRs to decrease control traffic to make shortest hop forwarding paths.

In [6] we can see a brief analysis and examples of MPR selection algorithms which may result in better or worse convergence times (depending on the number of nodes).

While the protocol is working each MPR node must maintain information about its neighbors which are called "Multipoint Relay Selector set". This information is obtained by the way of Hello messages. Broadcast messages are only to be retransmitted by MPRs and only if it has not received it yet. The set of nodes that are eligible for MPRs can change during the course of mobility scenarios which is continuously updated by the Hello messages.

Each OLSR packet has the same structure:



Packet Length		Packet Sequence Number
Message Type	Vtime	Message Size
Originator Address		
Time to Live	Hop count	Message Sequence Number
MESSAGE		
Message Type	Vtime	Message Size
Originator Address		
Time to Live	Hop count	Message Sequence Number
MESSAGE		

Figure 3 - The OLSR Packet format [7]

From the packet header some fields are important to our problem:

Packet Sequence Number: This field permits nodes to identify older messages and so realize what information is the most reliable from messages they have received.

Vtime: This field is important in the way it informs the node of the time the information contained in the packet is valid. This field can be changed but if it is to change the time between Hello messages has to be accordingly regulated. It is important to note that the higher the validity time, usually to avoid traffic flooding, the less degree of convergence can be achieved in the network.

Time to Live: This field contains the maximum number of hops a message can be retransmitted, it decreases any time the message is retransmitted. A message with TTL 1 should not be retransmitted. This field can limit flooding radius.

The format of the HELLO message is as follows:

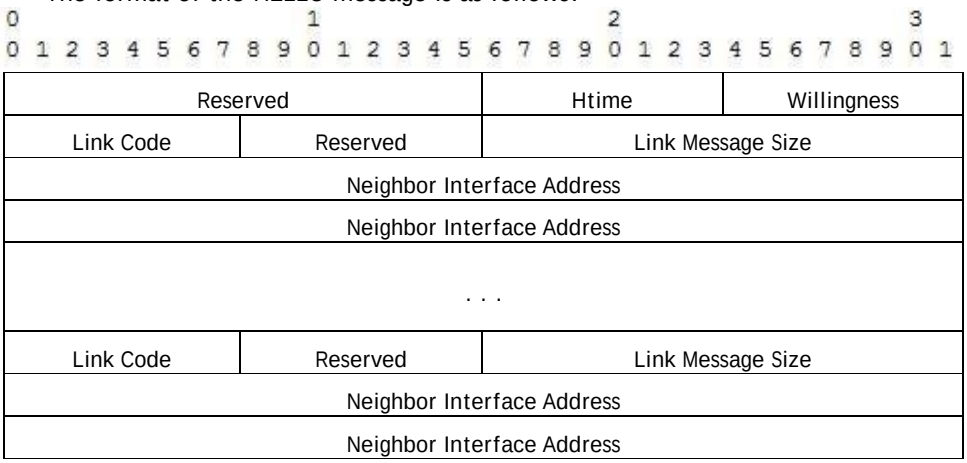


Figure 4 –The Hello message in OLSR [7]

From this message the fields important to our problem are:

HTime: This field is the time between Hello messages, which means the time that will pass before the next Hello message is sent from this particular neighbor.

Neighbor Interface Address: This field contains the address of an interface of a neighboring node. It permits the sender to obtain, through the answer to this message, its one and two hop neighbors to then select as mentioned before between the set of one hop neighbors the ones that provide best routes to the two hop neighbors.

The Hello message serve three tasks: link sensing, neighbor detection and MPR selection signaling.

The format of the TC message is as follows:

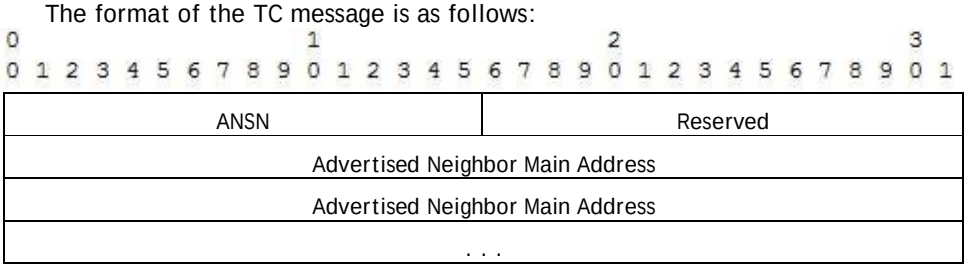


Figure 5 – The TC message in OLSR [7]

From this message the fields important to our problem are:

ANSN: A sequence numbered which allows for nodes to know if the messages received are the most recent or not.

Advertised Neighbor Main Address: This field provides receiving nodes the main address of neighboring nodes of the sender.

TC messages are sent by each MPR and contain neighbor state information that is retransmitted throughout the network, with the advantage of MPRs being the only nodes to retransmit them. Such messages contain the information that allows nodes to create routing tables so they must be refreshed accordingly to a TC_interval, maximum time between the updates from TC messages.

As it was mentioned earlier this is a proactive protocol, TC messages are sent proactively in a way that allows for routing tables to contain updated information before use which means it has a low time of route discovery. Routing overhead is not directly proportional to the number of routes inside the network but is influenced by the method of choice of MPRs. The times of validity of information and of maintenance of information from Hello and TC messages can be changed inside the messages which allows for a certain degree of flexibility, we have mentioned before that this flexibility will also compromise the degree of reliability of the information in the routing tables.

In relation to the method of selection of the MPR it is mentioned in [8] an underlying robustness problem regarding the fact that, due to this feature, under a seventy five percent of coverage from a given MPR to isolated nodes in a two hop neighborhood if one of the MPRs fails there will be a great probability that one of those nodes is not able to receive messages from the given node.

In terms of security there are 2 concerns with this protocol:

It is mentioned in [9] that to prevent ad hoc routing messages from modification, impersonation, replay attacks and more generally from all attacks related to wireless networks, integrity and authentication services are required. It is noticed that the confidentiality of messages is not required since routing messages are intended to all network nodes. For this purpose, cryptographic mechanisms are required, which means that some sort of certificate for authentication and digital signature for integrity is needed. And even so, integrity and authentication of routing messages are probably insufficient to prevent from all possible attacks. The example of dishonest nodes that can act attacks even once that they have been authenticated with valid certificates. The selfish and wormhole attacks which have been simulated and evaluated in that paper represent an example of such attacks. Such attacks have been referred to Byzantine attacks and concern the cases when authenticated nodes cannot be trusted and don't act as defined by the protocol specifications.

To summarize, OLSR can solve the problem of mobility, which means fulfilling the requisite of mobility between nodes and terminals and between nodes and that terminals are not responsible for signaling the network during mobility scenarios; the problem of automatic configuration, which means fulfilling the requisite of the automatic configuration of nodes; the flexibility problem, which means fulfilling the requisite of supporting different wireless technologies; nevertheless the protocol does not solve the scalability problem not being able to support a network of around ten thousand terminals and around one thousand and five

hundred nodes (MPRs) and default MPRs selection and time values for the variables on the messages would result in a big overhead,.

2.1.3. AODV

Specially designed for mobile ad-hoc networks, AODV is a distance vector reactive routing protocol that enables dynamic, self-starting multi-hop routing among mobile nodes participating in the network. [10]

As was mentioned, AODV is a reactive protocol, this means it does not maintain updated information of any routes but obtains quickly routes when activated. It offers mobility solution signaling nodes in the mobility scenario when links are broken so that they can correct the routes through which messages are sent.

One unique characteristic of the AODV is the use of destination sequence numbers for the route entries in the routing table, which results in loop-freedom that is easy to achieve computationally. This means that when a route is to be chosen, if there is more than one choice, it will be sent through the route with the highest sequence number.

AODV uses RREQ, RREP and RERR through UDP traffic to obtain the routes only when needed. When a terminal has connection communication with another one it does not use AODV (AODV works at control plane, not data plane); when it does not have such connection it works as follows: When a route is needed the source router sends a RREQ in broadcast with broadcast-id, source, destination, source sequence number, destination sequence number and hop count, this last one is incremented every time the RREQ is retransmitted, the message is then forwarded by intermediary nodes. A route can be determined when the message reaches the destination or a node with an updated route to the destination. An updated route means the route entry for the target destination has an associated sequence number as great as or greater than the one contained in the RREQ. The RREP is then sent to the source node, while caching the route to the source to make the RREP unicast from the destination and intermediary nodes. The source node uses the route to destination with the least hops. When a packet to a target destination is lost a link break is detected in the route and so the source node transmits a RERR message indicating that the destination is no longer reachable for that particular route. A list of precursors is used to enable the proper functioning of the RERR mechanism. This list contains neighbors that are likely to use current node as next hop in routes it “knows” and is usually filled up during the processing of a RREP message. Fields in a routing table entry in AODV are as follows [10]: Destination IP address, Destination Sequence Number, Valid Destination Sequence Number flag, other state and routing flags (e.g., valid, invalid, repairable, being repaired), Network Interface, Hop Count (number of hops needed to reach destination), Next Hop, Next Hop, List of Precursors, Lifetime (expiration or deletion time of the route). Sequence numbers guarantee loop

freedom, so, each node maintains its own sequence number. That number is changed in one of two situations: Immediately before a node originates a route discovery, it has to increment its own sequence number to prevent conflicts with previously established reverse routes towards the originator of a RREQ and also in the moment before a destination node originates a RREP in response to a RREQ, it has to update its own sequence number to the maximum between its current sequence numbers and the destination sequence number in the RREQ packet.

The RREQ message is as follows:

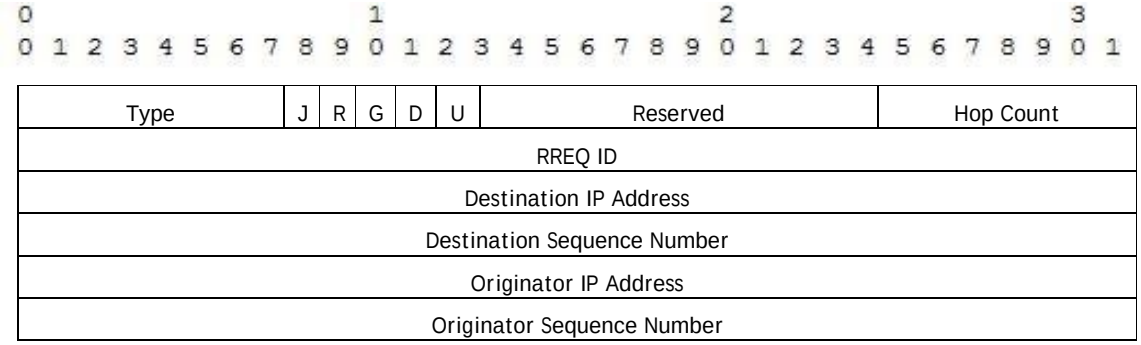


Figure 6 – The RREQ message in AODV [10]

Of these fields it is important to note the Hop count, which enables the originator to establish the number of hops necessary to achieve the node that is currently handling the message; the RREQ ID, which enables nodes to identify the RREQ in combination with the Originator IP Address; and the Originator Sequence Number, which informs of the sequence number to be used in the entry pointing towards the originator

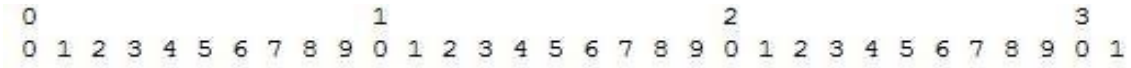
The RREP message is as follows:

Type	R	A	Reserved	Prefix Size	Hop Count
Destination IP Address					
Destination Sequence Number					
Originator IP Address					
Lifetime					

Figure 7 – The RREP message in AODV [10]

Of these fields it is important to note the Hop Count which enables both the originator and the destination of the number of hops between them, or in the case of Multicast, the number of hops between the closest member of the multicast to the originator; the destination sequence number, which allows for the updated information about the destination that the originator node needs; and the Lifetime which informs nodes receiving the RREP how long is the information in the message valid.

The RERR message is as follows:



Type	N	Reserved	Hop Count
Unreachable Destination IP Address			
Unreachable Destination Sequence Number			
Additional Unreachable Destination IP Address			
Additional Unreachable Destination Sequence Number			

Figure 8 - The RERR message in AODV [10]

Of these fields it is important to note the N flag which informs receiving nodes that the link is down but is in repair, so the route should not be deleted; and the unreachable destination sequence number so that receiving nodes can compare to other messages regarding such destinations.

The simplicity of AODV comes from the distance vector routing which is easy to implement and the consumption of bandwidth being lower than proactive protocols, and the negative aspect comes from the time to establish a connection, the possibility of one RREQ message obtaining several RREP messages can lead to some control overhead.

In a simulation [11] comparing DYMO to AODV (DYMO being the successor of AODV) it is clear that AODV Route Discovery Latency is greater since it uses Expanding Ring Search during Flooding. AODV has higher response time and routing overhead in the beginning than DYMO.

To summarize, AODV can solve the problem of mobility, which means fulfilling the requisite of mobility between nodes and terminals and between nodes but in current AODV scenarios terminals are responsible for signaling the network during mobility scenarios; it can also solve the problem of automatic configuration, which means fulfilling the requisite of the automatic configuration of nodes, the flexibility problem, which means fulfilling the requisite of supporting different wireless technologies; nonetheless the protocol cannot solve the scalability problem not being able to support a WiMetroNet scale network, nor the flooding problem, which means fulfilling the requisite of dealing with and not spreading broadcast traffic because it lacks the TTL field.

2.1.4. DYMO

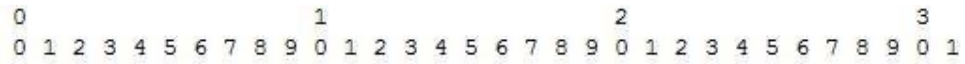
We will now consider a routing protocol that solves some mobility problems in routing in the WiMetroNet.

DYMO is a primarily reactive distance vector routing protocol although it can be proactive in maintaining routes updated. It is a successor of AODV in the way that it shares many similarities but it has been enhanced to improve routing scenarios. It is noted that DYMO enables reactive, multihop unicast routing among nodes participating in the same network [12]. It is used for mobile routers to “build” a mobile Ad Hoc network. This protocol uses some messages similar to the ones in AODV that are exchanged between routers to ensure network topology differing in a few fields and processing of such messages. Routers using DYMO who do not know of a particular destination send RREQ messages through every interface and routers between the source of the message and the destination use this message to establish routes to both end and passing routers because the RREQ message contains a list of all the nodes it has passed through. The destination sends RREP message as response to RREQ to the destination ensuring this way that destination has a route to the source, and equally every intermediate node learns the route to the target destination in case it was not yet known. The dynamics of this protocol comes from the maintenance of the routes. Routes are considered active as packets flow through the routes which means that whenever a packet is successfully forward that route lifetime is extended. Whenever a message is lost due to the end of hop limit, the route is considered broken, a route error

message is sent to the source router, and the route is deleted from the routing table, if new packets are to be sent to this node a new RREQ is used to establish a new route.

Some noticeable features about DYMO are: the fact that it uses sequence numbers (same as AODV) to ensure loop freedom and prevent the use of outdated routing information; it supports nodes using multiple interfaces in the MANET; it can perform in nodes that do not possess great memory features since only active routes are maintained in memory; it gives the ability to other nodes to perform route discovery if they are connected to such nodes, even in non-participant interfaces; it needs to be configured to obtain routes to a destination to provide to certain terminals; its routing algorithm can be used in a different network layer although that may require changes in the message formats.

We now consider the RREQ and RREP messages which are similar in format but differ in content:



IP Header

IP.SourceAddress				
IP.DestinationAddress = LL-MANET-ROUTERS				
IP TTL/HopLimit = 255				

UDP Header

Destination Port = MANET				

Message Header

RREQ-type	0 1 0 0 0 0 0 0	Msg-size = 23		
Msg-hoplimit				

Message TLV Block

Msg-tlv-block-size = 0				

Message Body – Address Block

Number Addrs=2	1 0 0 0 0	Rsv	HeadLength=3	Head
Head (cont.)			Target.Tail	Orig.Tail

Message Body - Address Block TLV Block

Tlv-block-size = 6		DYMOSeqNum-type	0 1 0 1 0 0	Rsv
Index-start = 1	Tlv-length = 2	Orig. Seq Num		

Figure 9 – The RREQ message in DYMO [12]

From these fields it is important to note that:

-Msg-hoplimit is a field where the number of hops the message can perform can be limited.
-Target.tail is a field indicating the address of the target node for this message, which in the RREQ is the destination and in the RREP it is the source.
-Orig.tail is the field indicating the address of originator node of this message AND a prefix, which in the RREQ is the source node and its prefix, and in the RREP it is the destination node address and prefix for which a RREP is being generated.

It is noticeable that there are a lot of fields that can be attached to the message to condense information. One important note about the creation of the RREQ messages is that whenever a DYMO is about to create a RREQ message it has to increment its own sequence number by one to ensure the information is considered updated.

We now consider the RERR message:

Figure 10 – The RERR message in DYMO [12]

Fields worth mentioning are:

-IP.DestinationAddress is the field that contains the address to which this message will be forwarded to; in unicast this means the NextHopAddress, the adjacent DYMO router on the specified route

-HopLimit is the field which contains the number of nodes this message can pass through;

-UnreachableNode.Address which is the field where the address and prefix of the unreachable node, this field can contain several addresses.

One important benefit in DYMO is the Hop count field that allows for greater control on traffic to limit the overuse of the bandwidth.

In a simulation in [11] comparing DYMO to AODV it was observed that Route Discovery Latency in DYMO is lower than AODV, that usually DYMO has the lowest response time. It possesses routing overhead at the beginning of a session and such routing overhead is higher than AODV's. It is also observed that flooding in DYMO doesn't surpass 10 hops, but there are no route discoveries when the path is known; it sends more routing traffic than AODV due to path accumulation. DYMO has higher route discoveries and bigger message sizes because of the use of REBlock attachments (as was seen earlier in the message format). It is concluded that performance in smaller networks is better than larger ones where routing traffic overhead is significantly increased.

To summarize, DYMO can solve the problem of mobility, which means fulfilling the requisite of mobility between nodes and terminals and between nodes and terminals being responsible for signaling the network during mobility scenarios; the flooding problem, which means fulfilling the requisite of dealing with and not spreading broadcast traffic because it has a Hop count field; it can also solve the problem of automatic configuration, which means fulfilling the requisite of the automatic configuration of nodes; the flexibility problem, which means fulfilling the requisite of supporting different wireless technologies; even so the protocol cannot solve the scalability problem not being efficient supporting a WiMetroNet scale network.

2.1.5. TRILL

TRILL is a layer 2 network protocol that provides routing in large networks by changing the layer 2 frame in order to provide more direct access between Rbridges. The use of Rbridges is necessary to provide a scalable solution using a link-state protocol. Each Rbridge possesses a nickname that is derived from the 6-octet IS-IS System ID. Rbridges learn each others nicknames through a dynamic nickname acquisition protocol.

TRILL establishes connection between any two Rbridges in the network forming tunnels, in which those two Rbridges are each end of the tunnel. This tunnel has the following characteristics:

Communication is obtained when a terminal wants to send a frame towards another terminal providing the Ethernet address, upon arriving to the first Rbridge in the border of the network (ingress Rbridge) it will encapsulate the frame to the nearest Rbridge of End station destination providing its nickname and the nickname of that Rbridge (egress Rbridge). Traffic between these Rbridges will use an outer Ethernet header in a way that Rbridges will retransmit the frame from one another until the arrival at the nicknamed egress Rbridge.

This is an example of a TRILL frame:

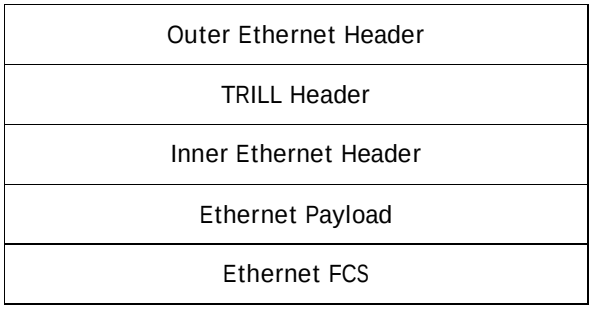


Figure 11 – The TRILL frame format [13]

The TRILL header is as follows:

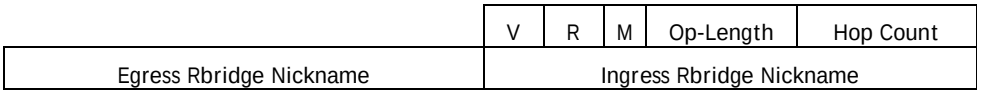
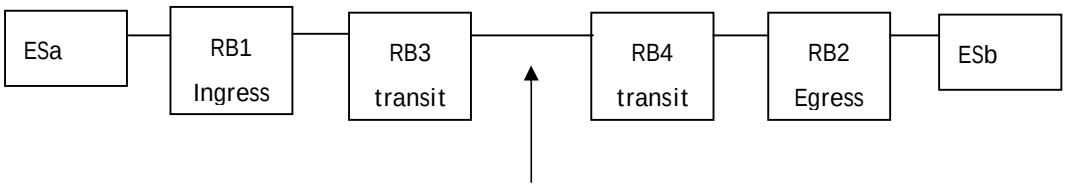


Figure 12 – The TRILL header [13]

A TRILL frame has 5 different headers, of which the top 3 are the most descriptive ones. The top header is link-specific (can be used in different technologies); the second header is the TRILL header that uses a hop count in order to provide loop mitigation and is independent of VLANs by using a separate VLAN tag, the nickname of the egress Rbridges (destination Rbridge, which is the Rbridge of the VLAN where the destination terminal is), and the nickname of the ingress Rbridge (source Rbridge, which is the Rbridge of the VLAN where the source terminal is).

TRILL needs Rbridges in bridged VLANs to run the IS-IS protocol to elect Rbridges on each bridged VLAN as the "Designated Rbridge" (DRB), which is the designated router similar from IS-IS protocol, and as in that protocol the DRB can give a pseudo node to the link, issue an LSP on behalf of that pseudo node, and specify to Rbridges on that link which VLAN is to be used for communications. The DRB is also responsible for encapsulating and decapsulating all traffic to and from the VLAN it is designated, or delegating that task to one of the Rbridges in the VLAN. The Rbridge that is responsible for forwarding in a VLAN must learn the address the terminals have in order to know to which VLAN forwarder Rbridge to send frames to and also the terminals in its own VLAN. It can learn such information in these ways according to [13]. For the links in its own VLAN it will learn from the source address from frames it receives, similar to bridges, or through a layer 2 explicit registration protocol; to learn the addresses in other VLANS it looks at the ingress Rbridge nickname in the header and VLAN and source addresses of the inner frame of TRILL frames it receives.

When a frame is in transit inside the network it has the following fields:



Outer Ethernet Header:

Outer Destination MAC address (RB4)	
Outer Destination MAC Address	Outer Source MAC Address
Outer Source MAC Address (RB3)	
Ethertype = C-Tag [802.1Q]	Outer.VLAN Tag Information

TRILL Header:

Ethertype = TRILL	V	R	M	Op-Length	Hop Count
Egress (RB2) Nickname	Ingress (RB1) Nickname				

Inner Ethernet Header:

Inner Destination MAC Address (ESb)	
Inner Destination MAC Address	Inner Source MAC Address
Inner Source MAC Address (ESa)	
Ethertype = C-Tag [802.1Q]	Inner.VLAN Tag Information

Payload:

Ethertype of Original Payload	
Original Ethernet Payload	

Frame Check Sequence:

New FCS (Frame Check Sequence)

Figure 13 – TRILL Data encapsulation [13]

To summarize, TRILL can solve the problem of flexibility, which means fulfilling the requisite of supporting different wireless technologies; the problem of flooding, which means fulfilling the requisite of dealing and not spreading broadcast traffic, because the header has a Hop Count field as was mentioned before; it does not solve the problem of scalability because it cannot support efficiently the number of terminals and Rbridges necessary to the network and it does have a very large header which would result in a serious problem of

overhead and bandwidth spending in the data plane; it does not solve the problem of mobility but the control and data plane are easily separated which means the data plane could be used with another routing protocol that allows for mobility; it also does not fulfill the dynamic configuration of terminals and nodes requisite.

2.1.6. Wireless Metropolitan Routing Protocol

The WMRP is a new routing protocol still under development by Gustavo Carneiro and Pedro Fortuna as part of their PhD works, that was created to be able to achieve certain goals other routing protocols couldn't resolve in a whole particularly for a WiMetroNet scenario.

One of the goals for the development of the WMRP was that it would be auto-configurable. This was a necessity because of the number of nodes that would participate in the network. It would be virtually impossible to alter code for every machine in the network.

Other goals are a standard requirement for any such networks: to support network and terminal mobility, to be able to scale up to ten thousand terminals and more than a thousand Rbridges, and a way to deal with broadcasts.

Given these goals, the protocol was created to work at a layer in the OSI Model connoted with the term "layer 2.5", which in turn allows for terminals to be 1 IP hop away from other terminals. While the base routing protocol works via periodic updates, as in most proactive link state routing protocols, certain optimizations are being developed to support mobility more efficiently.

2.1.6.1. Overview

Figure 14 summarizes the flow of information, its effects on the Rbridge's tables and what tables each message interacts with:

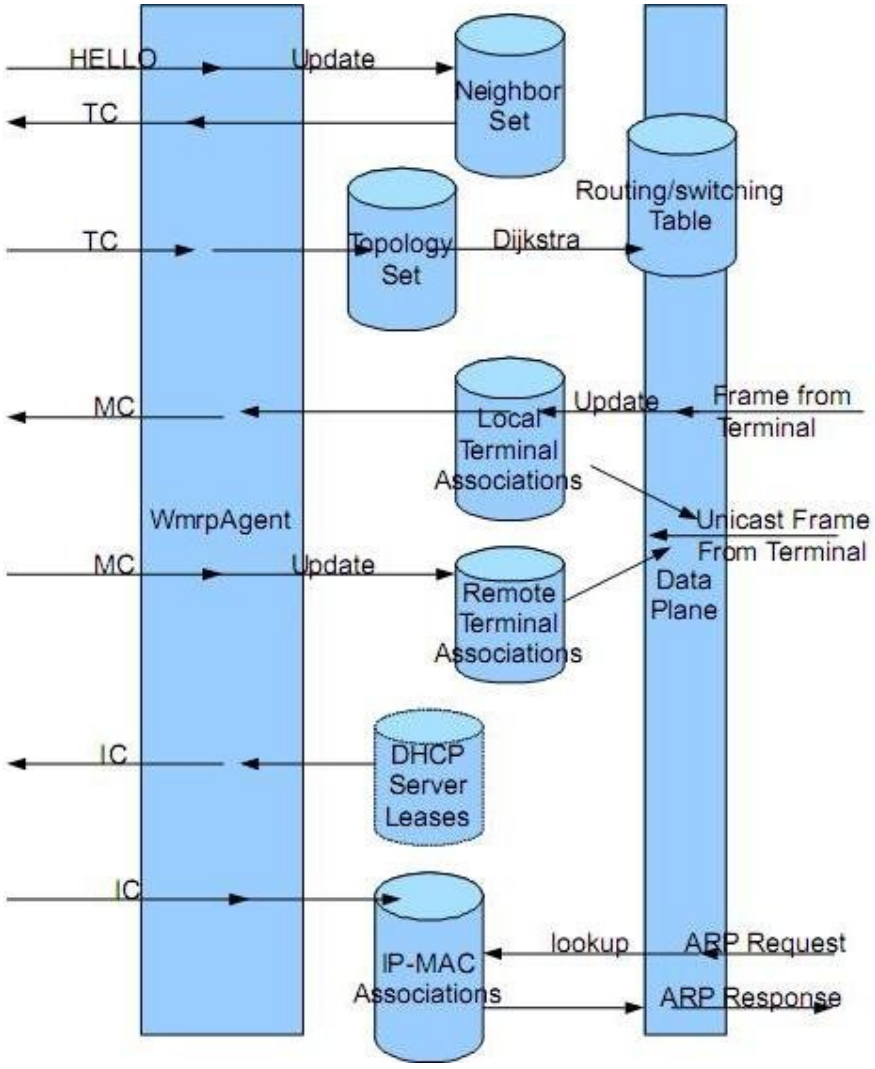


Figure 14 – Flow of information and messages between control (WMRPAgent) and data planes

As we can see from Figure 14 the protocol is based in the OLSR messages applied to an OSI model layer 2 environment. Each node has a different 20-bit ID. One way proposed to ensure that every node has a different Id is to build it based on one of the MAC addresses of the machine. Like OLSR, WMRP is a proactive protocol. Using Hello messages every node announces its existence to neighbor nodes this message is sent through each interface every two seconds. The information collected from these Hello messages can be hold in a table for six seconds. To be able to know about other nodes beside its neighbors the protocol uses TC messages that are sent by each node with a 255 TTL that only contain its neighbor node Id's; this way an Rbridge knows where all the other nodes are. With the information of different nodes Ids an Rbridge builds a Routing table containing a node Id as a label, the interface associated and the following Rbridge MAC address to reach it.

The WiMetroNet reference consists of a core made by Rbridges connected to each other and exterior terminals connected to the core via an access Rbridge. In this protocol, any contact made from the terminals would be filtered by the access Rbridges. Broadcast

traffic isn't forwarded to the core and each terminal's Mac address would be stored by the data plane in a LocalTerminalAssociation table. The information contained in this table would be passed periodically in a MC message that relays the information contained in the LocalTerminalAssociation table to other Rbridges with the source of the information (Originator node Id) that in turn use it to make a RemoteTerminalAssociation table. This way, if an Ethernet frame to a known MAC Address is received the node can insert the label corresponding to the corresponding node Id.

Edge Rbridges also filters DHCP traffic to store IP-MAC associations inside an IPMacAssociation table. Each Rbridge sends periodically this information in an IC message containing the information in the IPMacAssociation table which is then forwarded to each neighbor Rbridge. When a terminal needs to find a destination's address it sends an ARP request which is verified by the Ingress Rbridge in the IPMacAssociation table and the Rbridge will create a corresponding response to answer the ARP request if it finds the associated Mac address.

For transmission of messages a common header is used in the WMRP:

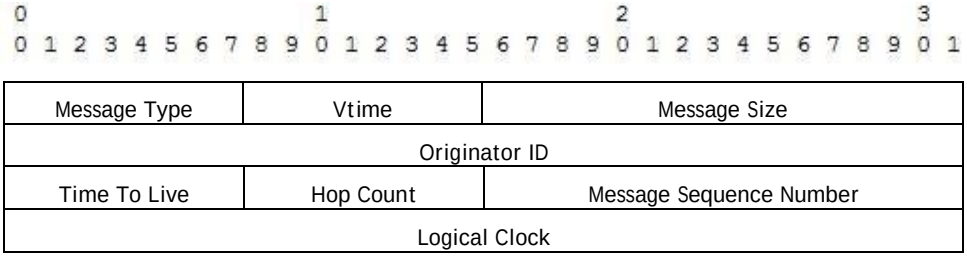


Figure 15 – WMRP header

This header possesses the following information about the underlying message: the Message Type field shows what kind of message is in the frame, Hello, TC, MC or IC; the Vtime is a field with information about how much time the information in this message is valid (it is not in seconds); Message Size is the size of message in bytes; the originator ID shows receiving nodes what is the node ID of the Rbridge that created the message, which in turn can be different from the ID of the last Rbridge to transmit the message; Time To Live is the number of times the message can be forwarded by any receiving Rbridge (TTL of 1 or 0 should not be forwarded); the Hop Count field is the number of times this message has been forwarded by any Rbridge in the network; the Message Sequence Number is a unique number that allows the detection of duplicate messages, when coupled with the Originator ID; finally the Logical Clock is the field in the frame that allows marshaling of events coming from different nodes.

2.1.6.2. The HELLO message

The Hello message in the WMRP is used only to detect neighbors:

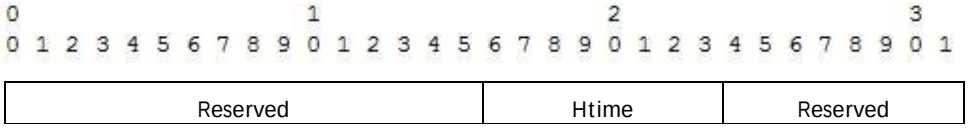


Figure 16 – WMRP Hello Message

Its only field is the Htime, or Hold time, which is the time for which information about the source of this Hello message should be stored in neighbor routing tables, default is 6 seconds. Since all the information about the source of the hello message is in the Header there is no need for further fields. Hello Messages are only intended for Neighbor nodes so TTL in all Hello messages is 1.

2.1.6.3. The TC message

TC messages are the backbone of the network. When sent they are transmitted in a radius fashion to provide Rbridges in the network with information about the presence of Rbridges that aren't directly connected.

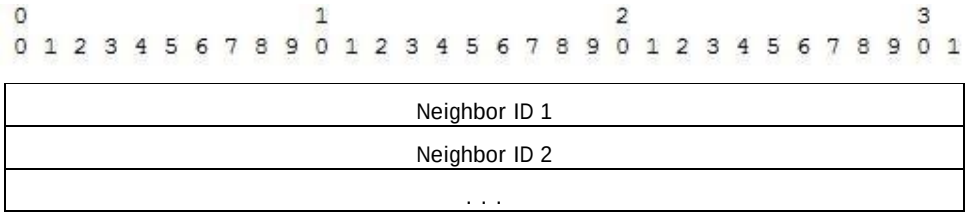


Figure 17 – WMRP TC Message

Every TC message sent has the node ID of the currently active neighbors of the Rbridge that created it. These messages always start with a 255 TTL but are never retransmitted by the same Rbridge twice.

2.1.6.4. The MC message

MC messages relay Mac Address information about active terminals in their neighborhood:

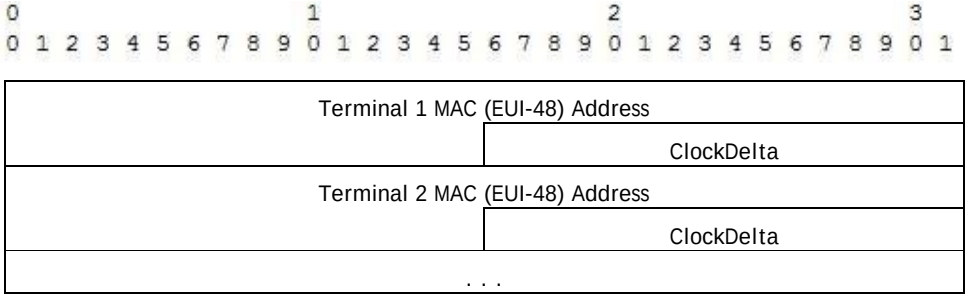


Figure 18 – WMRP MC Message

Every MC message possesses the entirety of its neighbors list of terminals although the message can be segmented into different frames because of MTU limitation. Each neighbor is shown as a 48 bits Mac Address with a timestamp of when it was last seen active in the network.

2.1.6.5. The IC message

IC messages are used to relay information about IP-MAC associations to other Rbridges. They are only sent by Rbridges connected to a DHCP server.

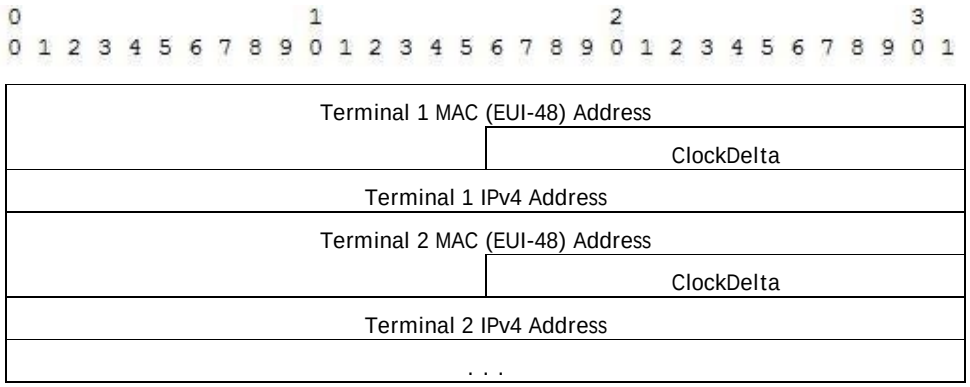


Figure 19 – WMRP IC Message

The IC message transmits both the Mac Address and the IP address of terminals in its neighborhood with the time information of when that terminal acquired an IP lease from a DHCP server which makes them valid for 60 seconds by default.

2.1.7. Summary

All in all, none of the initial protocols discussed can be used in whole for the network intended but they can be partially used to create a possible solution. MPLS would be chosen over TRILL because it can provide a better scalable solution for the network and the less overhead due to simple label switching with a smaller header, however it does not provide mobility support, but because it can easily be separated in a data plane and control plane it was chosen for the data plane. As for the routing protocol in the control plane, OLSR is the best candidate because even though the other protocols do possess mobility support they have very large time consuming route setups due to their reactive nature and do not scale for such a large network. OLSR has less overhead but still it consumes very large bandwidth to topology updates to ensure convergence which would be immense in such a large network as is the WiMetroNet. Rbridges provide use of the network in a lower layer to enable 1-IP hop away terminals with terminals maintaining their IP.

The WiMetroNet architecture borrowed ideas from TRILL, MPLS, and OLSR.

The WMRP protocol was chosen for the control plane because while being based in OLSR, using Hello and TC messages without the use of MPR's and being proactive, it possesses less overhead and uses Rbridges which allow for layer 2 routing functioning at a lower layer in the OSI Model. It also is under development to achieve mobility optimizations that would require it to have much less route updates than OLSR.

2.2. Network Simulator 3

This section will cover the use of the Network Simulator 3 (ns3) and its ability to emulate.

A simulating environment is one of the cheapest ways to draw conclusions about a model without actually having to assemble or produce any devices. Network simulation provides a malleable way to predict the behavior of a network. The use of mathematical models and the ability to change attributes to different network devices can be used to experiment with many different kinds of networks and possible outcomes of projects involving networks.

The simulator chosen to analyze and simulate the WMRP by the WiMetroNet team is the NS-3. This simulator was developed under the project nsnam, which is the acronym for network simulation and network animation, although it is not an evolution or a newer version of NS-2, a previous widely used network simulator. NS-3 is a discrete-event network simulator

for Internet systems, targeted primarily for research and educational use. It is Open Source Software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use [14]. It is written in C++ programming language with the option of resorting to python interfacing, for this aspect it has a python binding module.

The choice of the NS-3 is due to some key aspects that are important to this project. The scalability and extensibility is better supported than other network simulator. It provides the required scalability for the WiMetroNet requirements. It provides greater approximation to real machines and uses packets similar to real network packets, being bit by bit represented exactly the same way; it uses sockets API; it reuses kernel and application code, and it has memory optimization. One other aspect of the NS-3 is the possibility to use callback objects, function objects manipulated by value [14]. In recent studies comparing current simulators NS-3 was considered to have the best overall performance with low computational and less memory demands [15].

NS-3 supports software integration with other open source applications and also a closer approach to tracing and statistic requirements, through various trace sources with a statistical and data management framework. It also possesses an attribute system very well documented to provide greater malleability to simulations. Lastly but not least it provides emulation modes for virtualization and testbed integration.

It is regarding this last property of NS-3 that the work starts to be developed. Network emulation is a way to insert the simulator in a physical network to test and verify reactions to physical data. Emulation uses interfaces from a simulator and can send data to real devices and receive data from those devices to insert into the simulator. The emulation module in NS-3, ns3-emu module, was made for the ns3.3 version by Craig Dowell providing a way to use any machine that can support NS-3 to be able to introduce it in real scenarios.

The problem for our implementation is that the WMRP was developed under version NS-3.2 so there is a need to either backport (use the emulation software to function with the older version) or to achieve another way to emulate the WMRP.

3. Work description and analysis

3.1. Problem Characterization

In order to build a metropolitan area network such as WiMetroNet, which is considered by [1] as a large Mesh Network of moving Rbridges operating at layer 2.5 over heterogeneous wireless technologies, some aspects are to be considered. There are two different planes that needed to be implemented; one is the control plane, the routing protocol, the WMRP. The other is the data plane, the plane where data is addressed to the required terminal or server. The implementation was divided in two different planes and here we will address only the control plane, which is the one implemented.

Terminals access the network inside vehicles and at bus stops or train stations or tram stations. Each vehicle should support one access station, and the same applies to each bus stop or train station or tram station. Vehicles should be able to connect to the network via the stations access or some vehicular protocol.

Many large networks function properly even when a part of the network is non-functional. This is common in distributed systems. A wireless metropolitan network has the same concept; the network should be able to respond to different requests even when part of the network isn't working properly. The use of Rbridges as the access points to customers of the metropolitan transports means they are critical elements to guarantee connectivity with the server and the rest of the network. These Rbridges are to support both the data and the control plane. The structure of the network must be able to detect flaws so that when a node becomes unavailable there is always one node that connects to it from where the system can find the flaw and if possible correct it.

As a metropolitan transports network, the protocol used should be able to support every single customer in range of the system, which means the number of clients connected through the system, either for server connection or P2P traffic, will surely be around the thousands. The number of clients connected inside the network will raise problems of bandwidth and quality of service given that the access points and the Rbridges will be connected to the server ultimately by Ethernet which has a limited bandwidth (depending on

the technology chosen) as in the wireless technology bandwidth for the vehicular protocol and terminal access. Scalability problems will not be addressed in the work of this thesis.

As in all the wireless networks, security becomes an increasing concern to protect data and traffic inside the network there has to be a mechanism to prevent the system of packet sniffing and rogue nodes trying to enter the network. However, security issues are out of scope of this work.

Due to the mobility aspect of the network, the routing protocol will have to be dynamic enough to guarantee there is no loss in data but the exchange of information between Rbridges should not be enough to overflow the entire system.

Given that the WiMetroNet is intended to be used in real world scenarios, the intent of this implementation is exactly to provide information that the WMRP can work in similar scenarios meaning it can successfully be reproduced in physical devices. The control plane should mainly do three things: 1) build a Routing Table between Rbridges in the network, 2) receive and disseminate information regarding DHCP to build an IP-Mac Address association Table (this is to provide information to the data plane) and, 3) upon receiving Remote Mac Associations, to build a Remote Mac Association Table with Remote Macs and the route to reach them. In turn, the kernel space data plane will relate with the control plane in three things: 1) Local Terminal Mac Addresses structured in a table with a timestamp of when that Mac Address was last seen sending frames to the network to allow the sending of MC messages in the WMRP, 2) detecting and intercepting ARP Requests and based on the IP-Mac Address association Table provide the requesting terminal with a corresponding response in the form of an ARP Response, 3) upon receiving any unicast or multicast frames for other terminals use the Routing Table to create a tunnel to send the frame through the core (if needed) or simply bridging the frame to the required interface.

3.2. Methodology

These were the relevant aspects and choices made during development of the solution that lead to the final adopted solution.

The protocol chosen to implement for this thesis was the WMRP. This protocol was however still under research. Given the fact that there was only an implemented WMRP inside a simulating environment, to be able to reproduce this protocol in a physical environment there was a particular need to be able to interface the ns3 modules of the WMRP with physical scenarios. There was already such an implementation in the latest ns3, the ns3.3 version in the ns3-emu module, but the WMRP modules were all written for the ns3.2 version of the simulator and, as such, unable to utilize with the ns3-emu module. Therefore a different way to achieve emulation was needed. The routing agent in the

simulation had to be adapted to run in a real machine and to provide the required information to the user space to interact in the testbed scenarios.

The Packet module in ns3 had a few aspects that were relevant in the choice of the adopted solution. Packets created inside the ns3 are similar to real packets, the content of a packet buffer is bit for bit equal to a real network packet [14]; packets created with "dummy" bytes (static size with no data, or fewer than actual packet size) do not result in memory allocated for those "dummy" bytes, which results in less memory spent; packet class has easy methods for creating a simulating packet with real data, Create<Packet>(), and reading data from a simulating packet, PeekData().

3.3. Adopted Solution

The implementation of the WMRP was to be used in real testbed scenarios. In these testbed scenarios each device in the network is independent and can function properly connected to any network it can be connected to. Therefore, the testing scenarios were all made using one Rbridge alone inside the simulator in each machine. Ethernet interfaces of that machine would provide for interfaces with other Rbridges also connected through Ethernet connections.

Since WMRP runs at layer two in the OSI model inside NS-3, a choice was made to utilize Sockets using the SOCK_RAW type to continue working in that layer in the real world scenarios. Packet sockets are sockets used to receive and send packets at OSI model layer 2 that uses the "socket()" function with socket_family as AF_PACKET. The use of socket_type SOCK_RAW enables the construction of raw packets, packets that are passed to and from the network device without any changes to the packet data. Frames sent and received through these sockets have a known structure:

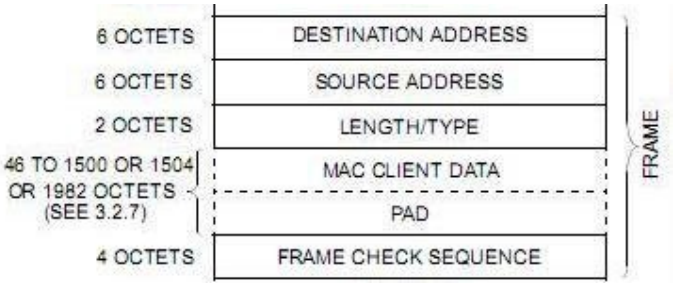


Figure 20 – Frame format in 802.3 type II [16]

The frames that are sent and that arrive at the created packet socket contain the destination address, the source address, the Ethertype and the WmrpMessage in the payload.

The destination and source addresses are the Mac Addresses that the source of the packet chooses: destination is always broadcast in the WMRP, and source is always the Mac Address of the Ethernet card from where the sender sent the frame. The Ethertype is a field similar to OSI model layer 3 ports but at layer 2, it identifies sub-protocols used at upper layers; for example, Internet Protocols use Ethertype 0x0800, ARP uses Ethertype 0x0806. The Ethertype chosen for WMRP was 0x1234. Inside the Ethernet frame payload is the WmrpMessage, which can contain one or more WMRP messages.

To be able to receive messages from outside NS-3 and sending of messages from inside NS-3 two different kinds of threads had to be implemented, receiving threads and the main thread. Having a socket in the receiving end and in the transmitting end implied that a receiving thread needed to schedule arriving frames. This way the Rbridge could continue working in the main thread without having to wait for other frames, and the main thread could transmit frames outside of the simulator whenever the Rbridge needed to send a WmrpMessage. To be able to receive frames in more than one interface and to detect in which interface the receiving frame arrived more than one receiving thread was created through the duplication of the methods that were created after pthread_create(). This was the easier solution because Global Variables in threads are not thread safe as such we could not create two threads using the same code without the interface address for which the thread was to receive frames from the network. The receiving threads communicated with the main thread by using the method “Simulator::ScheduleNow” to schedule the arrival of a WMRP message or messages in its Ethernet interface. The event scheduled calls method MainThreadReceive which transforms the frame payload from the received frame into a NS-3 packet to process. The method ScheduleNow is thread-safe, so it enables the scheduling of events from any arbitrary thread to the simulating thread. It schedules an event with expiring time as current time.

In the solutions with one and two threads the interfaces address was hard coded by choice because using the /proc files could result in segmentation fault. These files can change while tests are running, using netlinks to fetch interfaces from kernel space would require the implementation of a new kernel space/user space interface which is beyond the scope of this work.

As was mentioned in Sec. 2.2, NS-3 is a discrete event simulator in which every event that would happen in a real network is scheduled to occur in an ordered fashion and events occur immediately after each other since it provides with faster simulations. To be able to provide a scheduler that could deal with real time events the RealTimeScheduler was activated using the python code line:

```
ns3.GlobalValue.Bind("SimulatorImplementationType",
                     ns3.StringValue("ns3::RealtimeSimulatorImpl"))
```

The real time scheduler provides a real time implementation of the process of events. To provide with time consumption it uses sleep and busy-wait cycles until the actual time of the scheduled event is consistent with the simulator's time.

The choice for hard coded node Ids is due to simpler and smaller node Ids. The use of MAC based node Ids would result in bigger node Ids. This choice does not interfere with the implemented solution as nodes Ids always occupy 20 bits.

To be able to view changes in the routing tables every time the routing table changed a copy of the routing table was printed to a log file with the change and with the current time.

3.4. Code structure

The ns3 simulated the WMRP with an interface between native ns3 modules and the WmrpAgent, that interface is the WiMetroNet class.

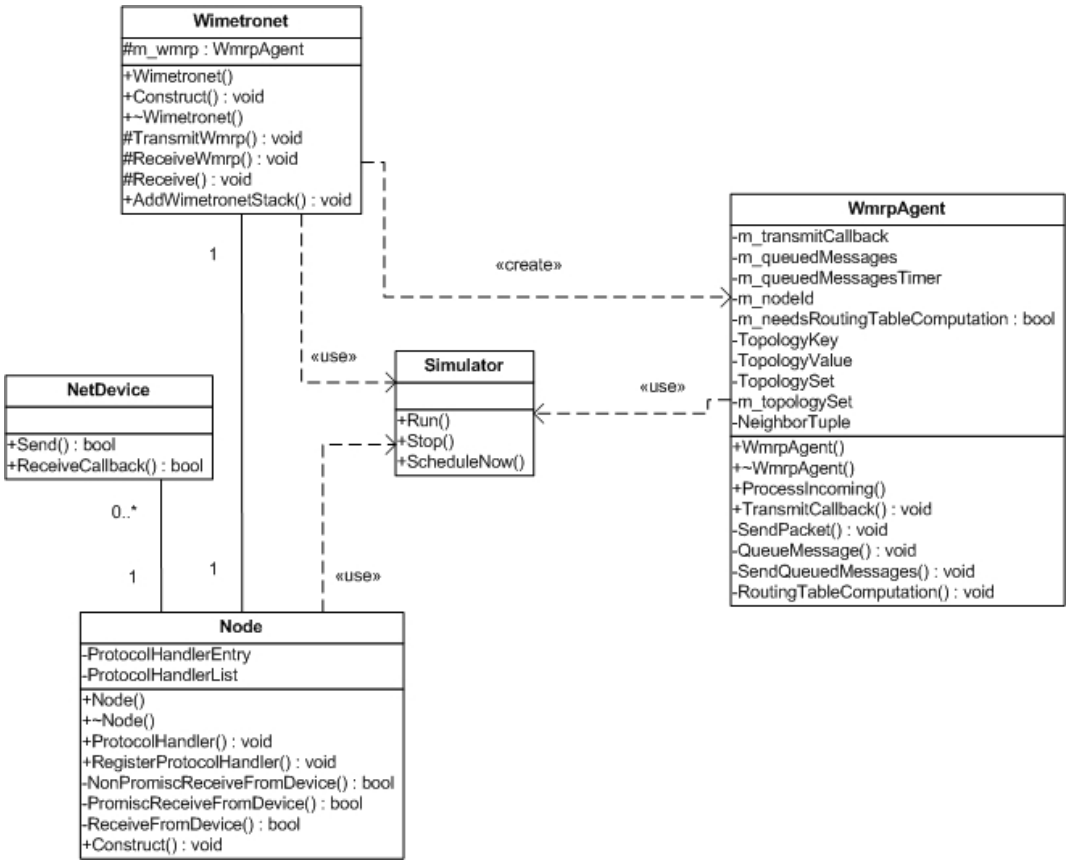


Figure 21 – Class diagram for WMRP in ns3

Figure 21 shows the association between classes in the ns3 to install the WMRP in an ns3 node. The WiMetroNet Class adds the protocol in the node with the method "AddWiMetroNetStack" it also creates a callback when adding a NetDevice to a node through the "RegisterProtocolHandler()" in Node class to immediately process packets that arrive at the simulated node.

The following diagram describes how a packet is sent inside the ns3:

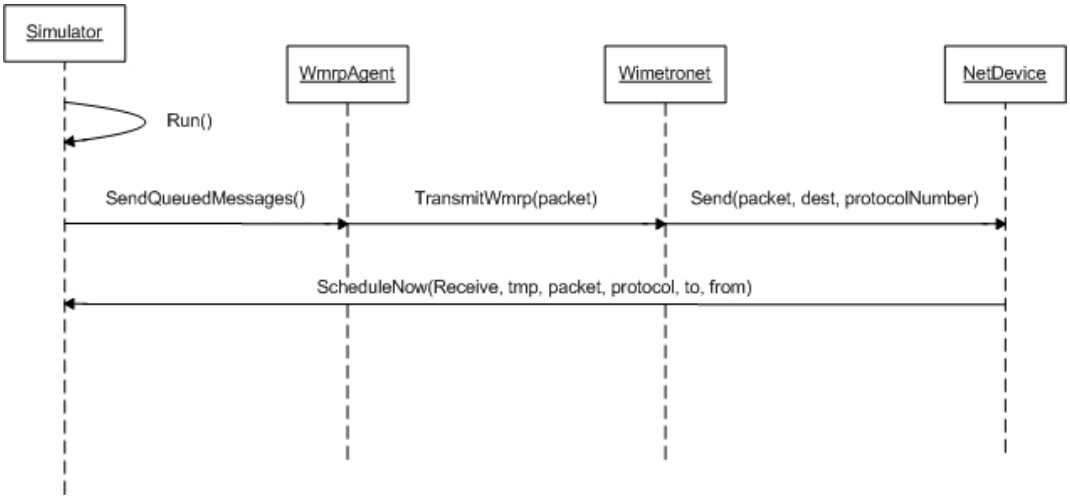


Figure 22 – Sequence diagram for sending a WMRP message in ns3

In Figure 22 the simulator starts running and will SendQueuedMessages when this event expires. The WmrpAgent will then construct packets with the queued messages and uses the "TransmitCallback" to call the "TransmitWmrp" method for each one. When the WiMetroNet is asked to transmit a WmrpMessage it uses the node's devices of where it is installed (transmission of packets is broadcasted from all the interfaces. Since these packets are sent inside the simulator the method "Send" in the netdevice simply schedules the receiving of the sent packet to the corresponding nodes.

Receiving a WmrpMessage in the simulator is described in the following diagram:

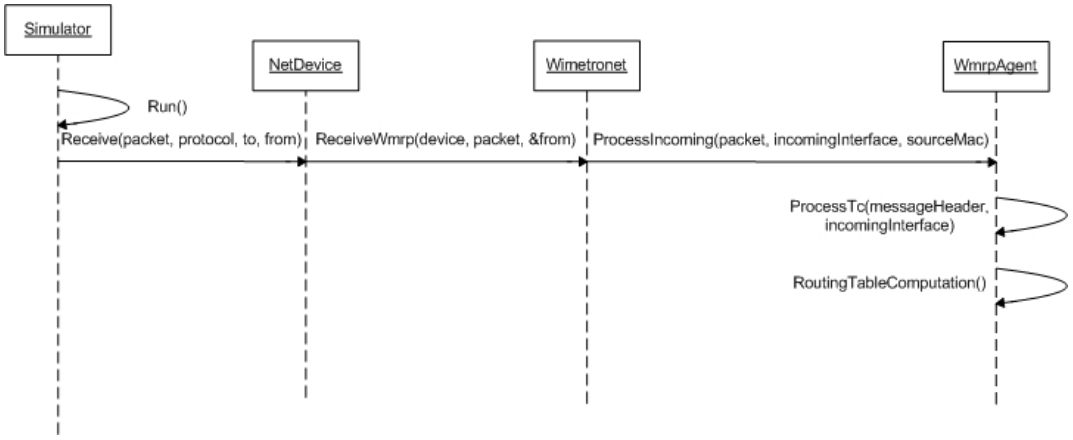


Figure 23 – Sequence diagram for receiving a WMRP message in ns3

To receive a message a node only has to wait for the simulator to expire the event of receiving the message. As is shown in Figure 23 the node receives in its NetDevice the package, for the purpose of the WMRP the sequence only shows what happens regarding the WMRP. Because the WiMetroNet had set a callback for receiving packets when the NetDevice received any package it receives it in the method "ReceiveWmrp". It receives the id of the device it received the packet from, the packet, and the Mac Address it came from. The WmrpAgent then processes the received message with the same attributes, processes the message (the described sequence shows "ProcessTc" but it can process Hellos, TC, MC and IC messages) and will compute the Routing table accordingly to the received message.

The code developed to implement the solution is based in sockets and threads with the use of the RealtimeSimulatorImpl class.

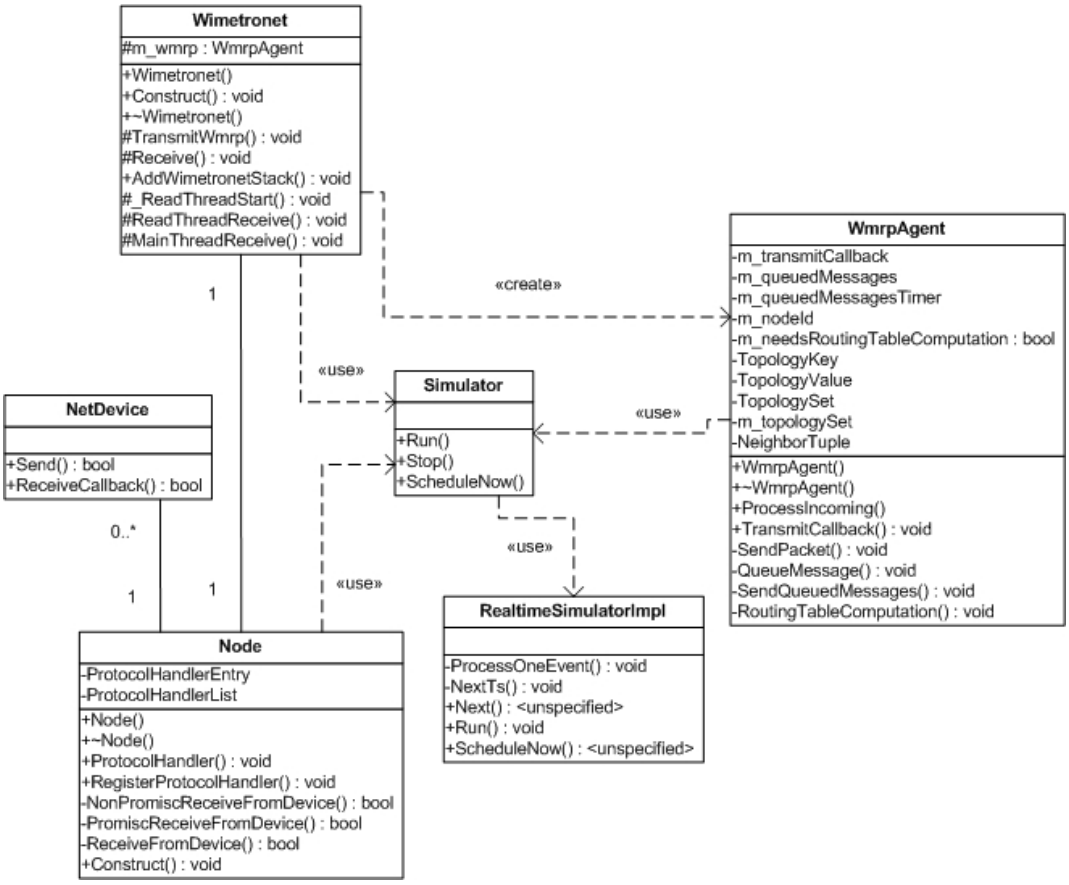


Figure 24 – Class diagram for physical implementation of WMRP in ns3

There are only 2 visible changes to the class diagram of the code structure: the use of the "RealtimeSimulatorImpl" class, which is the use of the Real Time Scheduler to enable the consumption of time in the Simulator, and three new methods inside the WiMetroNet class. The three new methods implement the thread part of the code. The use of sockets is invisible at this level; sockets are used in methods "TransmitWmrp" and "ReadThreadReceive".

To understand the workings of the code, a sequence diagram for the receiving of frames from the network, with the new methods and functions used, is displayed:

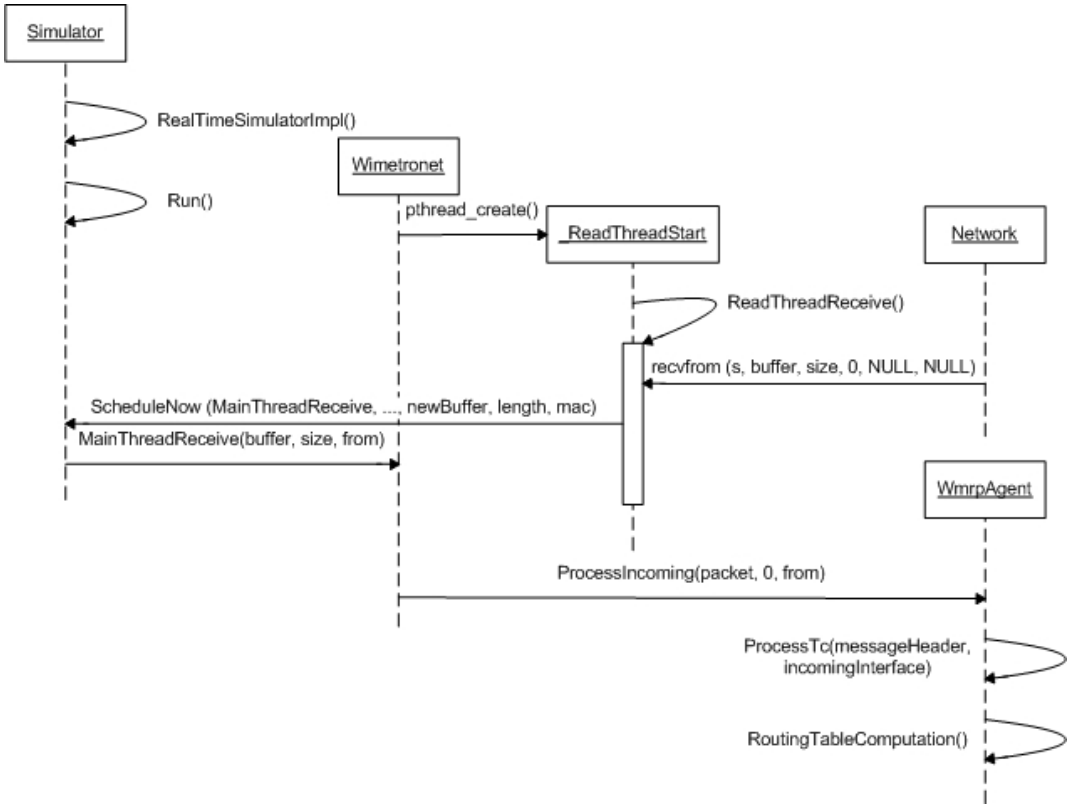


Figure 25 – Sequence diagram for receiving a physical WMRP message in ns3

The main process initializes WiMetroNet under the RealTimeSImulatorImpl with default attributes as before; but in the constructor one receiving thread for each interface is created with the pthread_create function as Figure 25 shows. The receiving thread triggers a continuous loop to receive frames from the network in a receiving socket through the interface it is bound to and Schedules to the simulator a method to process the arrival of new frames with the WMRP Ethertype to construct ns3 packages understandable to the WMRP. The main thread then proceeds to run in the WiMetroNet using the "MainThreadReceive" method. WMRP packets are then processed by the WmrpAgent in the same way as before the implementation.

For the transmission sequence a similar diagram was developed:

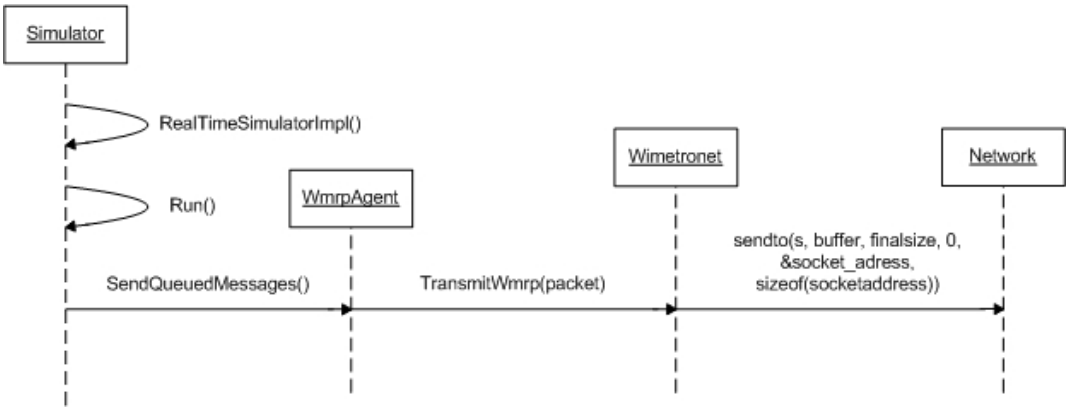


Figure 26 – Sequence diagram for transmitting a physical WMRP message in ns3

Figure 26 shows that the main thread implements the Real Time Scheduler with the call "RealTimeSImulatorImpl". The main thread will process scheduled events and transmit frames with WmrpMessages to the network whenever the WMRP timers get to 0. The queued messages are then processes in the WmrpAgent in the form of a WMRP packet which is sent to the WiMetroNet through the "TransmitWmrp" method, which in turn sends the packets in all the interfaces through multiple packet sockets that build the Ethernet Header with the use of the sockaddrl_ll structure.

4. Work Evaluation

To be able to analyze the results of the work developed it was necessary to design different tests. These tests can show which requirements were met and if indeed the protocol can respond to physical testbed scenarios. These tests were designed in increasing complexity of the topology to understand what problems would arise in any step. All of the tests were run under machines running NS-3 in Ubuntu 8.10 as the background operating system in an Intel Pentium 4 machine with 1.5 GB RAM. The capture of packets/frames was made using Wireshark 1.2, a packet sniffer.

4.1. First test

The first test was designed in a testbed scenario with two machines running the WMRP protocol inside ns3 in which those machines were connected by Ethernet to a switch in a home network. This is the most basic scenario in which one can analyze if the WMRP can indeed deal with the simplest topology. It also tested what happens to the Rbridge and its routing table when the connection is lost.

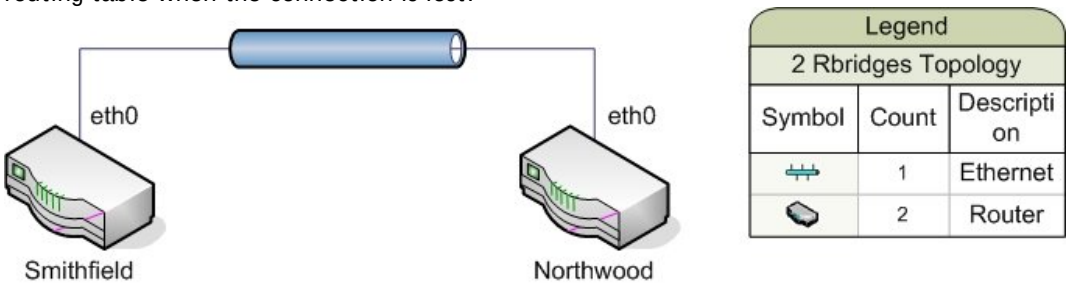


Figure 27 – First test topology

Table 1 – Test 1 Devices

Rbridge	Node ID	Eth0 Interface
Smithfield	9	00:0f:ea:f6:37:47
Northwood	1	00:13:8f:42:c5:67

In this test there was a continuous capture of frames in interface eth0 in the Northwood machine and the changes in the routing table in this machine as well (since it was a symmetric scenario where both machines ran the same program no logging was made in the Smithfield machine).

The following table shows the capture made in the interface eth0 during the testing scenario of the first test:

Table 2 – Test 1 Capture:

Msize = Message size, OID= originator ID (identifier), TTL= Time to live, HC=Hop Count, MSN=Message Sequence Number, LC=Logical Clock, NID=Neighbor ID. The Smithfield originated frames have white background, Northwood originated frames have gray background.

Packet Number	Time	Source	Info
1	0.000000	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=0 LC=0 HT=134
2	2.011913	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=3 LC=0 HT=134
3	2.710458	00:0F:EA:F6:37:47	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=0 LC=2 HT=134 TC Msize=20 OID=9 TTL=255 HC=0 MSN=0 LC=1 NID=1
4	3.035886	00:13:8F:42:C5:67	TC Msize=20 OID=9 TTL=254 HC=1 MSN=1 LC=2 NID=1
5	4.243894	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=4 LC=4 HT=134
6	4.801707	00:0F:EA:F6:37:47	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=3 LC=5 HT=134
7	5.152014	00:13:8F:42:C5:67	TC Msize=20 OID=1 TTL=255 HC=0 MSN=5 LC=4 NID=9
8	5.596541	00:0F:EA:F6:37:47	TC Msize=20 OID=1 TTL=254 HC=1 MSN=5 LC=4 NID=9
9	6.067219	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=7 LC=6 HT=134
10	6.791611	00:0F:EA:F6:37:47	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=4 LC=7 HT=134
11	7.688665	00:0F:EA:F6:37:47	TC Msize=20 OID=9 TTL=255 HC=0 MSN=5 LC=7 NID=1
12	7.915942	00:13:8F:42:C5:67	TC Msize=20 OID=9 TTL=254 HC=1 MSN=5 LC=7 NID=1 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=8 LC=9 HT=134
13	8.684210	00:0F:EA:F6:37:47	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=7 LC=10 HT=134
14	10.251939	00:13:8F:42:C5:67	TC Msize=20 OID=1 TTL=255 HC=0 MSN=9 LC=11 NID=9

			Hello Msize=20 OID=1 TTL=1 HC=0 MSN=11 LC=11 HT=134
15	10.679840	00:0F:EA:F6:37:47	TC Msize=20 OID=1 TTL=254 HC=1 MSN=9 LC=11 NID=9 Hello Msize=20 OID=9 TTL=1 HC=0 MSN=8 LC=13 HT=134
16	12.267888	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=12 LC=14 HT=134
17	12.875799	00:0F:EA:F6:37:47	TC Msize=20 OID=9 TTL=255 HC=0 MSN=9 LC=15 NID=1 Hello Msize=20 OID=9 TTL=1 HC=0 MSN=11 LC=15 HT=134
18	13.188055	00:13:8F:42:C5:67	TC Msize=20 OID=9 TTL=254 HC=1 MSN=9 LC=15 NID=1
19	14.203911	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=13 LC=17 HT=134
20	14.775600	00:0F:EA:F6:37:47	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=12 LC=18 HT=134
21	15.111898	00:13:8F:42:C5:67	TC Msize=20 OID=1 TTL=255 HC=0 MSN=14 LC=19 NID=9
22	15.375261	00:0F:EA:F6:37:47	TC Msize=20 OID=1 TTL=254 HC=1 MSN=14 LC=19 NID=9
23	16.115905	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=16 LC=19 HT=134
24	16.573997	00:0F:EA:F6:37:47	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=13 LC=21 HT=134
25	17.569895	00:0F:EA:F6:37:47	TC Msize=20 OID=9 TTL=255 HC=0 MSN=14 LC=21 NID=1
26	18.031938	00:13:8F:42:C5:67	TC Msize=20 OID=9 TTL=254 HC=1 MSN=14 LC=21 NID=1 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=17 LC=23 HT=134
27	18.865630	00:0F:EA:F6:37:47	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=16 LC=24 HT=134
28	20.048059	00:13:8F:42:C5:67	TC Msize=20 OID=1 TTL=255 HC=0 MSN=18 LC=25 NID=9 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=20 LC=25 HT=134
29	22.263893	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=21 LC=25 HT=134
30	24.479883	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=22 LC=25 HT=134
31	26.271911	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=25 LC=25 HT=134
32	28.496128	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=26 LC=25 HT=134
33	30.645738	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=29 LC=25

			HT=134
34	32.283881	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=30 LC=25 HT=134
35	34.299903	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=31 LC=25 HT=134
36	36.475901	00:13:8F:42:C5:67	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=34 LC=25 HT=134

Hello messages and TC messages were sent frequently from both machines in the right times (2 seconds for Hello and 5 seconds for TC), when the Smithfield machine stops sending frames the Northwood machine stops sending TC messages, it only sends Hello Messages to try to find the rest of the network.

Logs of the routing table verify what the changes were to the routing table.

Every time a new Routing Table was loaded the Routing table was printed in a predefined form. The Routing Table always begins empty:

()

Table 3 – Test 1 Routing table log.

Each line in the Routing Table contains: A node ID of a node in the network known by the device, the Mac Address of where to send a frame to reach that node, and the Ethernet interface from where to send it to arrive there

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
9	00:0f:ea:f6:37:47	0	02:58:26.98834

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
N/A	N/A	N/A	02:58:48.355649

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
N/A	N/A	N/A	02:58:56.358870

When the Northwood machine finds hello messages from node 9 (the Smithfield machine) it was added to the routing table, once this node lost connection (stopped sending Hellos and the hold time for the information finished) the Routing table became empty.

The medium time between the break of connection from node 9 and the detection in node 1 is approximately 6.513 seconds. The standard deviation is 0.2367. Since the

theoretical values show that the time required between the loss of a neighbour and the change in the routing table to be between 4 seconds (which is the time for two Hello messages not arriving to the Rbridge) and 6 seconds and 700 milliseconds (which is the combined time for 3 Hellos messages, half a second for the refreshment of the routing table, and 200 milliseconds for the signalling) the result shows a slightly high medium time for this interval.

4.2. Second and third tests

The second and third tests comprehend three machines running the WMRP protocol. Because there weren't enough Ethernet interfaces available for these tests they were made in a virtualization scenario using VirtualBox.

VirtualBox is a virtualization product designed for x86 (8086 Intel family) microprocessors made by Sun Microsystems, Inc. This virtualization product was chosen because not only being the sole professional solution for virtualization available as Open Source Software under the GNU General Public Licence, it also runs in hosts using many different Operating Systems (Windows, Linux, Macintosh and Open Solaris) and, more importantly, can support a very large number of guest operating systems.

Tests using VirtualBox used for each machine a memory of 256 Mb RAM, 1.6 GB hard drive and Ethernet cards PCnet-Fast III (Am79C973), the information about each device and choice for node ID can be seen in the following table.

Table 4 – Tests 2 and 3 Devices and node IDs.

Rbridge	Node ID	Eth0 Interface	Eth1 interface
2	5	08:00:27:82:38:f1	08:00:27:82:e4:4e
3	9	08:00:27:93:b2:47	08:00:27:ff:a2:21
4	1	08:00:27:25:fc:a6	08:00:27:d9:ed:00

4.2.1. Second test

The second test comprehends three virtual machines connected via two virtual Ethernet connections in which only one of the machines is directly connected to the other two. It can provide better understanding of how, if capable, the routing tables are constructed with the Hello and TC messages when some Rbridges aren't directly connected to each other. There was also a break in the connection from one of the Rbridges connected to the center one to observe the changes of the routing tables that remained connected.

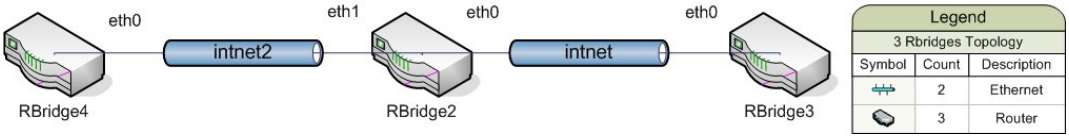


Figure 28 – Second test topology

In this test the connection *intnet* was severed, all routing tables changes were logged for comparing purposes but only frames in eth0 interface of the Rbridge4 machine were captured to see the difference in Hello and TC messages coming from the right side of the intnet2 network.

The following table shows the capture using wireshark made in the interface eth0 of the Rbridge4 during the testing scenario of the second test:

Table 5 – Test 2 Capture:

Msize = Message size, OID= originator ID (identifier), TTL= Time to live, HC=Hop Count, MSN=Message Sequence Number, LC=Logical Clock, NID=Neighbor ID. The Rbridge2 originated frames have white background, Rbridge4 originated frames have gray background.

Packet Number	Time	Source	Info
1	0.000000	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=0 LC=0 HT=134
2	4.114301	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=3 LC=0 HT=134
3	7.078047	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=0 LC=2 HT=134 TC Msize=20 OID=5 TTL=255 HC=0 MSN=1 LC=2 NID=1
4	8.610196	08:00:27:25:fc:a6	TC Msize=20 OID=5 TTL=254 HC=1 MSN=1 LC=2 NID=1
5	10.174736	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=3 LC=2 HT=134
6	13.830040	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=4 LC=2 HT=134
7	15.939311	08:00:27:82:e4:4e	TC Msize=20 OID=9 TTL=254 HC=1 MSN=1 LC=6 NID=5
8	16.114924	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=4 LC=6 HT=134

			TC Msize=20 OID=9 TTL=253 HC=2 MSN=1 LC=6 NID=5
9	16.534651	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=7 LC=8 HT=134
10	18.955104	08:00:27:25:fc:a6	TC Msize=20 OID=9 TTL=255 HC=0 MSN=5 LC=9 NID=5
11	19.349439	08:00:27:82:e4:4e	TC Msize=20 OID=9 TTL=254 HC=1 MSN=5 LC=9 NID=5
12	20.683584	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=8 LC=11 HT=134
13	22.137868	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=7 LC=12 HT=134
14	23.854870	08:00:27:82:e4:4e	TC Msize=24 OID=5 TTL=255 HC=0 MSN=9 LC=14 NID=9 NID=1 Hello Msize=20 OID=5 TTL=1 HC=0 MSN=11 LC=14 HT=134 TC Msize=20 OID=9 TTL=254 HC=1 MSN=5 LC=12 NID=5
15	24.936035	08:00:27:25:fc:a6	TC Msize=24 OID=5 TTL=254 HC=1 MSN=9 LC=14 NID=9 NID=1 TC Msize=20 OID=9 TTL=253 HC=2 MSN=5 LC=12 NID=5
16	27.357993	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=12 LC=15 HT=134
17	28.454239	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=8 LC=17 HT=134
18	30.406632	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=13 LC=18 HT=134
19	31.983813	08:00:27:82:e4:4e	TC Msize=24 OID=5 TTL=255 HC=0 MSN=14 LC=18 NID=9 NID=1
20	33.743396	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=16 LC=18 HT=134
21	36.302382	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=17 LC=18 HT=134
22	36.397335	08:00:27:25:fc:a6	TC Msize=20 OID=1 TTL=255 HC=0 MSN=9 LC=19 NID=5 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=11 LC=19 HT=134 TC Msize=24 OID=5 TTL=254 HC=1 MSN=14 LC=18 NID=9 NID=1
23	36.863061	08:00:27:82:e4:4e	TC Msize=20 OID=1 TTL=254 HC=1 MSN=9 LC=19 NID=5
24	38.122444	08:00:27:82:e4:4e	TC Msize=20 OID=5 TTL=255 HC=0 MSN=18 LC=21 NID=1 Hello Msize=20 OID=5 TTL=1 HC=0 MSN=20 LC=21 HT=134
25	38.937156	08:00:27:25:fc:a6	TC Msize=20 OID=5 TTL=254 HC=1 MSN=18 LC=21 NID=1
26	40.641872	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=12 LC=24 HT=134
27	43.028805	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=13 LC=24

			HT=134
28	44.145914	08:00:27:25:fc:a6	TC Msize=20 OID=1 TTL=255 HC=0 MSN=14 LC=21 NID=5
29	44.748858	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=16 LC=24 HT=134

Rbridge2 (node ID 5) forwarded TC messages from Rbridge3 (node ID 9) allowing Rbridge4 (node ID 1) to know the existence of Rbridge3 (node ID 9) in the network. Frame 24 shows that Rbridge3 (node ID 9) is no longer connected to the network.

Table 6 – Test 2 Routing table log in Rbridge3.

Each line in the Routing Table contains: A node ID of a node in the network known by the device, the Mac Address of where to send a frame to reach that node, and the Ethernet interface from where to send it to arrive there

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:38:f1	0	01:45:38.642903

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:82:38:f1	0	01:45:50.954198
5	08:00:27:82:38:f1	0	01:45:50.954348

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:82:38:f1	0	01:45:55.323607
5	08:00:27:82:38:f1	0	01:45:55.331047

The Routing table's first change was the addition of node 5 (neighbor node), having received the information that node 1 was a neighbor of node 5 added it to the Routing Table in "06-25-2009 01:45:50.954198".

Table 7 – Test 2 Routing table log in Rbridge2.

Each line in the Routing Table contains: A node ID of a node in the network known by the device, the Mac Address of where to send a frame to reach that node, and the Ethernet interface from where to send it to arrive there

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
N/A	N/A	N/A	01:45:31.625827

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
N/A	N/A	N/A	01:45:44.747870

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
9	08:00:27:93:b2:47	0	01:45:46.674377

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
9	08:00:27:93:b2:47	0	01:45:47.649935

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:45:49.905686
9	08:00:27:93:b2:47	0	01:45:49.905915

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:46:03.245083

The Routing Table of Rbridge2 (node ID 5) showed current available connections in the network because it was the only Rbridge with connections to every other device of the network. It was neighbor to both Rbridge4 (node ID 1) and Rbridge3 (node ID 9). The final loss of connection to Rbridge3 (node ID 9) is consistent with what was expected from the caused break in the second part of the test.

Table 8 – Test 2 Routing table log in Rbridge4.

Each line in the Routing Table contains: A node ID of a node in the network known by the device, the Mac Address of where to send a frame to reach that node, and the Ethernet interface from where to send it to arrive there

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:45:39.658263

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:45:47.746945

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:45:55.612927
9	08:00:27:82:e4:4e	0	01:45:55.613510

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:46:10.345234

The Routing Table of Rbridge4 (node ID 1) maintained a constant route to Rbridge2 (node ID 5) its neighbor from the start of detecting it until the end of the test. It detected the presence of Rbridge3 (node ID 9) behind Rbridge2 (node ID 5) therefore the next Hop Address is the same as to Rbridge2 (node ID 5) and after the break the Rbridge3 (node ID 9) was removed due to lack of detection.

The medium time between the break of connection from node 9 and the detection in node 1 is approximately 14.32828 seconds. The standard deviation is 0.203521. Since the theoretical values show that the time required between the loss of a hidden node connected to a neighbour and the change in the routing table to be between 10 seconds (which is the

time for two TC messages not arriving to the Rbridge) and 21 seconds and 700 milliseconds (which is the combined time for 3 TC messages, 15 seconds, and the 6 seconds and 700 milliseconds is the maximum time the node between them takes to know the loss of the disconnected node, which is its neighbor) the result shows an average medium time for this interval.

4.2.2. Third test

The third test comprehends three virtual machines connected via three virtual Ethernet connections in which every machine is connected to the other two machines in the network and the break in one of the Ethernet connections to verify the mobility requisite of the implementation, if indeed a change in the network can be translated into different routes when there is still connections available.

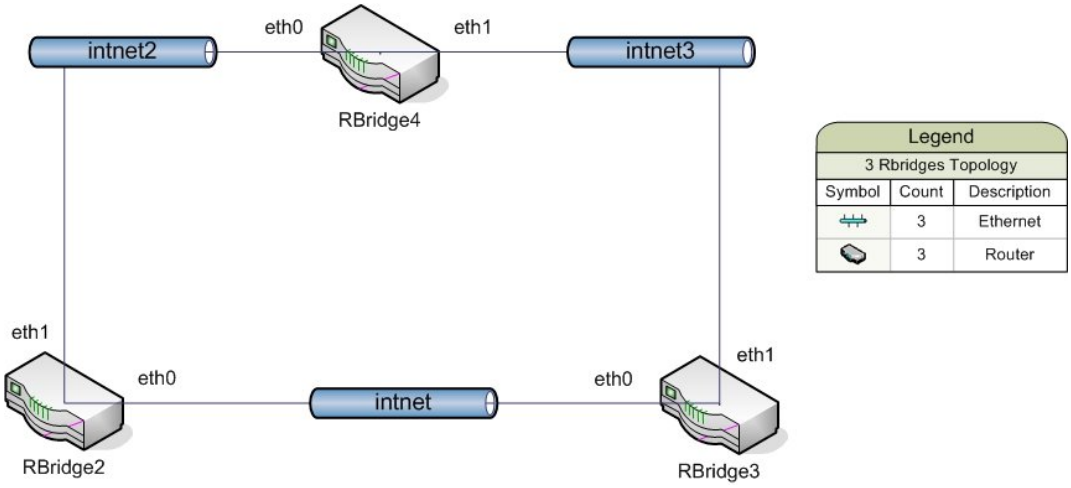


Figure 29 – Third test topology

In this test the connection intnet3 was severed, all routing tables changes were logged for comparing purposes and frames in both eth0 and eth1 interface of the Rbridge4 machine were captured.

The following table shows the capture made in the interface eth1 of the Rbridge4 during the testing scenario of the third test:

Table 9 – Test 3 Capture eth1:

Msize = Message size, OID= originator ID (identifier), TTL= Time to live, HC=Hop Count, MSN=Message Sequence Number, LC=Logical Clock, NID=Neighbor ID. The Rbridge3 originated frames have white background, Rbridge4 originated frames have gray background.

Packet Number	Time	Source	Info
1	0.000000	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=0 LC=0 HT=134
2	4.409386	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=3 LC=0 HT=134
3	7.980688	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=4 LC=0 HT=134
4	8.151470	08:00:27:ff:a2:21	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=0 LC=2 HT=134 TC Msize=20 OID=9 TTL=255 HC=0 MSN=1 LC=2 NID=1
5	8.819252	08:00:27:d9:ed:00	TC Msize=20 OID=9 TTL=254 HC=1 MSN=1 LC=2 NID=1
6	9.643697	08:00:27:d9:ed:00	TC Msize=20 OID=1 TTL=255 HC=0 MSN=5 LC=4 NID=9
7	10.578921	08:00:27:ff:a2:21	TC Msize=20 OID=1 TTL=254 HC=1 MSN=5 LC=4 NID=9
8	11.829415	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=7 LC=4 HT=134
9	12.263142	08:00:27:ff:a2:21	Hello Msize=20 OID=9 TTL=1 HC=0 MSN=3 LC=5 HT=134
10	15.681861	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=8 LC=6 HT=134
11	19.289710	08:00:27:d9:ed:00	TC Msize=20 OID=1 TTL=255 HC=0 MSN=9 LC=6 NID=9 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=11 LC=6 HT=134
12	20.632689	08:00:27:ff:a2:21	TC Msize=20 OID=1 TTL=254 HC=1 MSN=9 LC=6 NID=9 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=4 LC=9 HT=134
13	22.156036	08:00:27:ff:a2:21	TC Msize=24 OID=5 TTL=254 HC=1 MSN=1 LC=11 NID=1 NID=9
14	22.282702	08:00:27:d9:ed:00	TC Msize=24 OID=5 TTL=254 HC=1 MSN=1 LC=11 NID=1 NID=9 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=12 LC=13 HT=134
15	26.937926	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=13 LC=15 HT=134
16	28.927837	08:00:27:d9:ed:00	TC Msize=24 OID=1 TTL=255 HC=0 MSN=14 LC=16 NID=9 NID=5
17	29.740109	08:00:27:d9:ed:00	TC Msize=24 OID=5 TTL=254 HC=1 MSN=5 LC=16 NID=1 NID=9
18	31.395668	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=16 LC=17 HT=134

19	35.637818	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=17 LC=19 HT=134
20	37.284560	08:00:27:d9:ed:00	TC Msize=24 OID=9 TTL=253 HC=2 MSN=5 LC=14 NID=1 NID=5
21	39.728989	08:00:27:d9:ed:00	TC Msize=24 OID=5 TTL=254 HC=1 MSN=9 LC=22 NID=1 NID=9
22	47.482493	08:00:27:d9:ed:00	TC Msize=24 OID=9 TTL=253 HC=2 MSN=9 LC=26 NID=1 NID=5 TC Msize=24 OID=5 TTL=254 HC=1 MSN=14 LC=28 NID=1 NID=9
23	48.871052	08:00:27:d9:ed:00	TC Msize=20 OID=1 TTL=255 HC=0 MSN=18 LC=32 NID=5 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=20 LC=32 HT=134
24	53.945844	08:00:27:d9:ed:00	TC Msize=20 OID=9 TTL=253 HC=2 MSN=14 LC=33 NID=5
25	56.538811	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=21 LC=37 HT=134 TC Msize=24 OID=5 TTL=254 HC=1 MSN=18 LC=36 NID=1 NID=9
26	60.054562	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=22 LC=40 HT=134
27	63.100710	08:00:27:d9:ed:00	TC Msize=20 OID=1 TTL=255 HC=0 MSN=23 LC=42 NID=5
28	63.754806	08:00:27:d9:ed:00	TC Msize=20 OID=5 TTL=254 HC=1 MSN=23 LC=43 NID=1
29	66.235791	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=25 LC=44 HT=134
30	69.589539	08:00:27:d9:ed:00	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=26 LC=44 HT=134
31	71.512182	08:00:27:d9:ed:00	TC Msize=20 OID=1 TTL=255 HC=0 MSN=27 LC=44 NID=5 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=29 LC=44 HT=134

Both Rbridge4 and Rbridge3 successfully forward TC messages from Rbrige2 allowing both Rbridge4 and 3 to find a second route to reach Rbridge2, although the default choice will always be the one with the least amount of hops. There is a successful topology build, even after the break in connection from interface eth1 in the Rbridge3, because of TC messages coming to Rbridge4 from the other interface of Rbridge4 which are forwarded through this interface showing that Rbridge3 found a different route to Rbridge4.

The following table shows the capture made in the interface eth0 of the Rbridge4 during the testing scenario of the third test:

Table 10 – Test 3 Capture eth0:

Msize = Message size, OID= originator ID (identifier), TTL= Time to live, HC=Hop Count, MSN=Message Sequence Number, LC=Logical Clock, NID=Neighbor ID. The Rbridge2 originated frames have white background, Rbridge4 originated frames have gray background.

Packet number	Time	Source	Info
1	0.000000	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=0 LC=0 HT=134
2	4.438086	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=3 LC=0 HT=134
3	8.009412	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=4 LC=0 HT=134
4	8.582730	08:00:27:82:e4:4e	TC Msize=20 OID=9 TTL=254 HC=1 MSN=1 LC=2 NID=1
5	8.847956	08:00:27:25:fc:a6	TC Msize=20 OID=9 TTL=253 HC=2 MSN=1 LC=2 NID=1
6	9.672343	08:00:27:25:fc:a6	TC Msize=20 OID=1 TTL=255 HC=0 MSN=5 LC=4 NID=9
7	10.196738	08:00:27:82:e4:4e	TC Msize=20 OID=1 TTL=254 HC=1 MSN=5 LC=4 NID=9
8	11.852306	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=7 LC=4 HT=134
9	15.710449	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=8 LC=6 HT=134
10	19.318416	08:00:27:25:fc:a6	TC Msize=20 OID=1 TTL=255 HC=0 MSN=9 LC=6 NID=9 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=11 LC=6 HT=134
11	19.787199	08:00:27:82:e4:4e	TC Msize=20 OID=1 TTL=254 HC=1 MSN=9 LC=6 NID=9
12	21.384108	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=0 LC=11 HT=134 TC Msize=24 OID=5 TTL=255 HC=0 MSN=1 LC=11 NID=1 NID=9
13	22.311332	08:00:27:25:fc:a6	TC Msize=24 OID=5 TTL=254 HC=1 MSN=1 LC=11 NID=1 NID=9 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=12 LC=13 HT=134
14	24.403601	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=3 LC=14 HT=134
15	26.954840	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=13 LC=15 HT=134
16	27.383417	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=4 LC=14 HT=134
17	28.956303	08:00:27:25:fc:a6	TC Msize=24 OID=1 TTL=255 HC=0 MSN=14 LC=16 NID=9 NID=5
18	29.295592	08:00:27:82:e4:4e	TC Msize=24 OID=5 TTL=255 HC=0 MSN=5 LC=16 NID=1 NID=9

			TC Msize=24 OID=1 TTL=254 HC=1 MSN=14 LC=16 NID=9 NID=5
19	29.766431	08:00:27:25:fc:a6	TC Msize=24 OID=5 TTL=254 HC=1 MSN=5 LC=16 NID=1 NID=9
20	30.980408	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=7 LC=17 HT=134
21	31.424317	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=16 LC=17 HT=134
22	34.815287	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=8 LC=18 HT=134
23	35.666524	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=17 LC=19 HT=134
24	35.976626	08:00:27:82:e4:4e	TC Msize=24 OID=9 TTL=254 HC=1 MSN=5 LC=14 NID=1 NID=5
25	37.313211	08:00:27:25:fc:a6	TC Msize=24 OID=9 TTL=253 HC=2 MSN=5 LC=14 NID=1 NID=5
26	38.036184	08:00:27:82:e4:4e	TC Msize=24 OID=5 TTL=255 HC=0 MSN=9 LC=22 NID=1 NID=9 Hello Msize=20 OID=5 TTL=1 HC=0 MSN=11 LC=22 HT=134
27	39.743345	08:00:27:25:fc:a6	TC Msize=24 OID=5 TTL=254 HC=1 MSN=9 LC=22 NID=1 NID=9
28	41.079397	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=12 LC=25 HT=134
29	44.099026	08:00:27:82:e4:4e	TC Msize=24 OID=9 TTL=254 HC=1 MSN=9 LC=26 NID=1 NID=5 Hello Msize=20 OID=5 TTL=1 HC=0 MSN=13 LC=28 HT=134
30	45.351326	08:00:27:82:e4:4e	TC Msize=24 OID=5 TTL=255 HC=0 MSN=14 LC=28 NID=1 NID=9
31	47.447792	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=16 LC=31 HT=134
32	47.511181	08:00:27:25:fc:a6	TC Msize=24 OID=9 TTL=253 HC=2 MSN=9 LC=26 NID=1 NID=5 TC Msize=24 OID=5 TTL=254 HC=1 MSN=14 LC=28 NID=1 NID=9
33	48.897997	08:00:27:25:fc:a6	TC Msize=20 OID=1 TTL=255 HC=0 MSN=18 LC=32 NID=5 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=20 LC=32

			HT=134
34	49.582459	08:00:27:82:e4:4e	TC Msize=20 OID=1 TTL=254 HC=1 MSN=18 LC=32 NID=5
35	51.146930	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=17 LC=35 HT=134
36	53.269632	08:00:27:82:e4:4e	TC Msize=20 OID=9 TTL=254 HC=1 MSN=14 LC=33 NID=5
37	53.974547	08:00:27:25:fc:a6	TC Msize=20 OID=9 TTL=253 HC=2 MSN=14 LC=33 NID=5
38	54.712034	08:00:27:82:e4:4e	TC Msize=24 OID=5 TTL=255 HC=0 MSN=18 LC=36 NID=9 NID=1 Hello Msize=20 OID=5 TTL=1 HC=0 MSN=20 LC=36 HT=134
39	56.567484	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=21 LC=37 HT=134 TC Msize=24 OID=5 TTL=254 HC=1 MSN=18 LC=36 NID=9 NID=1
40	57.642930	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=21 LC=38 HT=134
41	60.082888	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=22 LC=40 HT=134
42	61.221605	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=22 LC=41 HT=134
43	63.115663	08:00:27:82:e4:4e	TC Msize=20 OID=5 TTL=255 HC=0 MSN=23 LC=41 NID=1
44	63.123307	08:00:27:25:fc:a6	TC Msize=20 OID=1 TTL=255 HC=0 MSN=23 LC=42 NID=5
45	63.482336	08:00:27:82:e4:4e	TC Msize=20 OID=1 TTL=254 HC=1 MSN=23 LC=42 NID=5
46	63.776270	08:00:27:25:fc:a6	TC Msize=20 OID=5 TTL=254 HC=1 MSN=23 LC=41 NID=1
47	64.807689	08:00:27:82:e4:4e	Hello Msize=20 OID=5 TTL=1 HC=0 MSN=25 LC=43 HT=134
48	66.249403	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=25 LC=44 HT=134
49	69.618252	08:00:27:25:fc:a6	Hello Msize=20 OID=1 TTL=1 HC=0 MSN=26 LC=44 HT=134
50	71.540853	08:00:27:25:fc:a6	TC Msize=20 OID=1 TTL=255 HC=0 MSN=27 LC=44 NID=5 Hello Msize=20 OID=1 TTL=1 HC=0 MSN=29 LC=44 HT=134

Both Rbridge4 and Rbridge2 successfully forward TC messages from Rbrige3 allowing both Rbridge4 and 2 to find a second route to reach Rbridge3, although the default choice will always be the one with the least amount of hops. There is a successful topology establishment to Rbridge3, even after the break in connection from interface eth1 in the

Rbridge3, because of TC messages from Rbridge3 forwarded by the Rbridge2. Rbridge2 continues to maintain connections to both Rbridge3 and 4 as demonstrated by frames 29 and 38.

Table 11 – Test 3 Routing table log in Rbridge3.

Each line in the Routing Table contains: A node ID of a node in the network known by the device, the Mac Address of where to send a frame to reach that node, and the Ethernet interface from where to send it to arrive there

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
N/A	N/A	N/A	01:32:26.169576

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:d9:ed:00	1	01:32:35.623706

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:d9:ed:00	1	01:32:47.720752
5	08:00:27:82:38:f1	0	01:32:47.720997

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:d9:ed:00	1	01:33:00.757004
5	08:00:27:82:38:f1	0	01:33:00.757169

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:82:38:f1	0	01:33:10.900208
5	08:00:27:82:38:f1	0	01:33:10.937878

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:82:38:f1	0	01:33:15.488318
5	08:00:27:82:38:f1	0	01:33:15.488542

The Routing table's first change was the addition of node 1 (first active neighbor node) and soon after that node 5. By 06-25-2009 01:33:10.900208 the node lost connection in interface 1 (eth1) to node 1 and having received TC messages from node 5 supporting connection to node 1 it was added as a new route to node 1.

Table 12 – Test 3 Routing table log in Rbridge2.

Each line in the Routing Table contains: A node ID of a node in the network known by the device, the Mac Address of where to send a frame to reach that node, and the Ethernet interface from where to send it to arrive there

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
N/A	N/A	N/A	01:32:26.525902

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
9	08:00:27:93:b2:47	0	01:32:33.451404

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:32:35.431156
9	08:00:27:93:b2:47	0	01:32:35.431385

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:32:54.578861
9	08:00:27:93:b2:47	0	01:32:54.579015

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:32:56.51540
9	08:00:27:25:fc:a6	1	01:32:56.51796

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:32:58.495165

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:32:56.51540
9	08:00:27:25:fc:a6	1	01:32:56.51796

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:33:00.949042
9	08:00:27:25:fc:a6	1	01:33:00.949247

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:33:02.638959
9	08:00:27:93:b2:47	0	01:33:02.639315

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:93:b2:47	0	01:33:11.183303
9	08:00:27:93:b2:47	0	01:33:11.183452

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:33:14.750119
9	08:00:27:93:b2:47	0	01:33:14.750344

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:33:18.371559
9	08:00:27:93:b2:47	0	01:33:18.371793

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
1	08:00:27:25:fc:a6	1	01:33:26.275455

The Routing table in Rbridge2 sustained a lot of unexpected changes, probably due to lack of Hello messages (lack of Hello messages from node 1 to node 5 can be seen in table 10). When received information from neighboring nodes the Routing table maintained a different route to node 1 and 9, each from a different interface. At "01:32:56.51796" node 9 was perceived as being behind node 1, for this to have happened it can only mean that node 9 didn't maintain a constant transmission of Hellos to node 5 (through intent). It also shows that node 9 was temporarily lost at "01:32:58.495165", since node 5 didn't even see it as behind node 1 although it is quickly recovered at "01:33:00.949247"; at that time node 9 was still connected to node 1. A correct route to node 9 is achieved at "01:33:02.639315" but lack of Hello messages from node 1 change route to node 1 at "01:33:11.183303" which is only regained correctly at "01:33:14.750119", the final loss of connection to node 9 is expected due to closing of simulations.

Table 13 – Test 3 Routing table log in Rbridge4.

Each line in the Routing Table contains: A node ID of a node in the network known by the device, the Mac Address of where to send a frame to reach that node, and the Ethernet interface from where to send it to arrive there

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
9	08:00:27:ff:a2:21	1	01:32:34.291356

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:32:47.477773
9	08:00:27:ff:a2:21	1	01:32:47.550487

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:33:00.72346
9	08:00:27:82:e4:4e	0	01:33:00.79060

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:33:02.59618
9	08:00:27:82:e4:4e	0	01:33:02.59843

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:33:19.349256
9	08:00:27:82:e4:4e	0	01:33:19.349406

Destination Node ID	Next Hop		Time added
	Mac	Interface ID	
5	08:00:27:82:e4:4e	0	01:33:29.221587

The Routing table in Rbridge4 (node ID 1) sustained only expected changes. In the beginning of the test it maintained a different route to each node, one in each interface. When having lost connection to node 9 it changed the route to the other interface, through node 5.

The medium time between the break of connection from node 9 and the detection in node 1 is approximately 5.852011 seconds. The standard deviation is 2,165355. Since the theoretical values show that the time required between the loss of a neighbour and the change in the routing table to be between 4 seconds (which is the time for two Hello messages not arriving to the Rbridge) and 6 seconds and 700 milliseconds (which is the combined time for 3 Hellos messages, half a second for the refreshment of the routing table, and two hundred milliseconds for the signalling) the result shows an average medium time for this interval.

4.3. Discussion of Results

The first test suggests that the Rbridge can function independently correctly and that changes to neighboring topology are noticed and those changes are respectively made to the Routing table. The medium time taken to detect the loss of connection with real machines was a little high but considering both machines were in a domestic environment with several other machines accessing the network through the switch which was directly connected to the router, and the machines tested were running other processes it is understandably high.

The second test shows that Rbridges are detected even outside the immediate neighborhood and problems or changes in those Rbridges are detected and result in changes in the network Rbridges. The time to detect the loss of connection from the two machines not directly connected was well within the boundaries of the theoretical predictions and envisions a possibility to scale well.

The third test shows that mobility scenarios are detectable and result in changes in the network Rbridges that display correct Route configuration upon topology changes. The time taken for detection was well within boundaries although it showed a great deviation but since all machines were directly connected it was not surprising.

These tests are but a fraction of all tests made to the network and all tests provided the same output. No tests were made for scalability purposes because of the lack of material for such implementations. VirtualBox can only provide up to 8 fully functional Ethernet interfaces in its virtual machines, further increase in devices in the network would result in host malfunction.

5. Conclusions

5.1. Revision of the work developed

The purpose of this implementation was to produce a solution that would validate the viability of the WMRP in real life scenarios. There were two main objectives achieved:

-The functionality of the WMRP was verified, in all the tests topology changes were detected and the Routing Table was corrected to represent them;

-The implementation of an open solution that can provide for a connection with the kernel space upon integration with the data plane, Routing Table is accessible to kernel space, the IP-Mac Association Table and the Local Mac Table can be built upon additions to this solution that in turn would simply add a greater number of messages sent by the WMRP.

5.2. Relevant contributions and results

There were two main contributions: the first was the integration of the simulation code with the use of “packet raw” sockets, threads and the real time scheduler which enabled the emulation; the second contribution was the validation of the WMRP in a testbed scenario subjecting machines using the WMRP through increasing complexity tests.

The implementation contribution positively allowed for the testing stages, without real frames in the network the tests could not have been made. Although the fact it used hard coded information about the devices and used duplicate code for each interface regarding the methods in which the reading thread ran, resulted in a hard to port code for each machine. Further work would have to be applied in this case.

The results from the tests suggest that WMRP is functional in testbed scenarios, which means, it can be transported for real network scenarios and that Rbridges in a network using it will adapt correctly to topology changes, even unexpected ones.

The use of Virtual machines provided with a poor solution for real machines because of lack of memory for each machine (256 Mb RAM) which resulted in a series of missed Hellos in the network which in turn resulted in unexpected topology changes in Rbridges that were connected.

5.3. Future Work

To be able to build the required WiMetroNet there is a need to integrate this solution with the data plane to provide a fully functional network using terminals and servers. The final solution would have to integrate both control plane and data plane.

To provide with automatic configuration of Rbridges there are some parts of the code that need to be changed from hard coding: the declaration of node IDs has to be based in Mac Address to provide different node IDs in the entire network; to be able to detect Ethernet interfaces and its Mac Addresses a new solution using netlink has to be developed [17] or file reading (ifcfg-eth) from the /proc files that have network information; and to allow the use of multiple threads without recurrent copying the code there needs to be developed a solution of turning global references into private references [18]

To be able to reproduce exactly the WiMetroNet, this protocol has to be used with wireless interfaces. Testing it in wireless interfaces is the next obvious step.

References

[1]Manuel Ricardo et all, "WiMetroNet-Scalable Wireless Network for Metropolitan Transports", 2008 Porto.

[2]Jong-Moon Chung, "Analysis of MPLS traffic engineering", http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=952816, August 2008, Lansing, MI, USA. Last access in 06/February/09

[3]José Ruela and Manuel Ricardo, "MPLS – Multiprotocol Label Switching", 2008, Porto, Portugal.

[4]Rick Gallaher, "An introduction to MPLS", http://www.convergedigest.com/Bandwidth/archive/010910TUTORIAL-rgallaher1.htm, 10 September 2001. Last access in 22/February/2009

[5]Anthony Busson, Nathalie Mitton, Eric Fleury, "An analysis of the MPR selection in OLSR", http://www.lri.fr/~fragile/IMG/ppt/OLSR-Fragile_AussoisMars05_leBon.ppt, March 2005, ARES INRIA/INSA de Lyon. Last access in 06/February/2009

[6]A.Qayyum, L.Viennot, A. Laouiti, "Multipoint relaying: An efficient technique for flooding in mobile wireless networks" Submitted to the 35th Annual Hawaii International Conference on System Sciences (HICSS'2001).

[7]T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)", http://www.ietf.org/rfc/rfc3626.txt, October 2003. Last access in 20/November/2008.

[8]Anthony Busson, Nathalie Mitton, Eric Fleury, "An analysis of the MPR selection in OLSR" http://hal.inria.fr/docs/00/07/05/39/PDF/RR-5468.pdf, January 2005, INRIA de Lyon. Last access in 06/February/2009

[9]Alia Fourati, Hakim Badis, and Khaldoun Al Agha, "Security Vulnerabilities Analysis of the OLSR Routing Protocol", http://www.lri.fr/~alia/Publis/ICT05.pdf, 2005. Last access in 06/February/2009

[10]C. Perkins and E. Belding-Royer, "Ad hoc On-Demand Distance Vector (AODV) Routing" http://www.ietf.org/rfc/rfc3561.txt, July 2003, California. Last access in 26/November/2008

[11]Koojana Kuladintihi, Carmelita Görg, "DYMO implementation in OPNET simulator", submitted to IETF-63 Paris, http://www.comnets.uni-bremen.de/~koo/kuladinithi-manet-ietf63.ppt, 4 August 2005. Last access in 06/February/2009

[12]I. Chakeres and C. Perkins, "Dynamic MANET On-demand (DYMO) Routing", http://tools.ietf.org/html/draft-ietf-manet-dymo-16, 5 December 2008. Last access in 20/February/2009.

[13]Radia Perlman et all, "Rbridges: Base Protocol Specification (TRILL)", http://www.ietf.org/internet-drafts/draft-ietf-trill-Rbridge-protocol-11.txt, 7 January 2009, Brocade. Last access in 22/February/2009.

[14]Various authors, "The ns-3 network simulator. Project Summary", http://www.nsnam.org/docs/proposal/summary.pdf, 2006-2009. Last access in 06/February/2009

[15]Elias Weingärtner, Hendrik vom Lehn and Klaus Wehrle, "A performance comparison of recent network simulators", http://ds.informatik.rwth-aachen.de/publications/2009/2008-06-Weingaertner-ICC-NetworkSimulatorComparison, Proceedings of the IEEE International Conference on Communications 2009 (ICC 2009), Dresden, Germany, IEEE. Last access in 26/June/2009

[16]IEEE Std 802.3™-2008: Part 3: Carrier sense multiple access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications

[17]"NETLINK(3)", http://linux.die.net/man/3/netlink, Linux Programmer's Manual, 2008-08-08

[18]Dave Marshall, "Threads: Basic Theory and libraries" http://www.cs.cf.ac.uk/Dave/C/node29.html, 1/5/1999 Cardiff School for Computer Science Last access in 26/June/2009

[19]Todd Litman, "Evaluating Public Transit benefits and Costs", http://www.vtpi.org/tranben.pdf, 2008, Victoria, Canada. Last access in 22/February/09

[20]Victoria Transport Policy Institute, "Transit Oriented Development", http://www.vtpi.org/tdm/tdm45.htm, August 2008, Victoria, Canada. Last access in 22/February/2009

[21]Jure Leskovec, "Dynamics of Large Networks", http://videolectures.net/sep08_leskovec_tdef/, September 2008, Carnegie Mellon University, Pittsburgh, Pennsylvania, U.S.A. Last Access in 22/February/2009

[22]Mark Steyvers, Joshua B. Tanenbaum, "The Large-Scale Structure of Semantic Networks", http://web.mit.edu/cocosci/Papers/03nSteyvers.pdf, 2005, Cognitive Science Society. Last access in 22/February/2009

[23]Journal/Magazine Gregory G. Finn, "Routing and Addressing Problems in Large Metropolitan-Scale Internetworks", http://eric.ed.gov/ERICDocs/data/ericdocs2sql/content_storage_01/0000019b/80/1c/4e/45.pdf, submitted to ISI Research report, March 1987, Washington D.C. Last access in 6/February/2009

[24]Rishi Srivatsavai et all, "OpenSolaris Project: Rbridge (IETF TRILL) support". http://opensolaris.org/os/project/Rbridges/, Last access in 22/November/2008.

[25]Wagner Meira Jr, "Redes Metropolitanas de Alta Velocidade", http://www.rnp.br/newsgen/9911/rmav.html, November 1999. Last access 06/February/2009

[26]Network Instruments LLC All Rights Reserved, "Network Instruments Observer version 12 features", http://www.networkinstruments.com/products/observer/new/3.html, 2007. Last access in 06/February/2009

[27]Dionísio Smith Nunes, Jonas Silva de Andrade, Marcelo Ribeiro Bazílio, "Simulação de rede metropolitana usando OPNE: Um estudo de caso", http://www.cci.unama.br/margalho/portaltcc/tcc2005/PDF/014.pdf, 2005. Last access 06/February/2009

[28]Edgard Jamhour, "Tecnologias sem fio para redes metropolitanas", http://www.ppgia.pucpr.br/~jamhour/Download/pub/Outros/Redes%20Mesh%20Metropolitanas.doc, August 2006. Last access in 6/February/2009