



UNIVERSITEIT•STELLENBOSCH•UNIVERSITY
jou kennisvennoot • your knowledge partner

Using ad hoc wireless networks to enable intelligent transport
systems: The design and analysis of the TH(O)RP routing
protocol

by

Daniel Weich Morrison



*Thesis presented in partial fulfilment of the requirements for the degree
of Master of Science in Engineering at the University of Stellenbosch*

Department of Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa

Supervisor: Dr R. Wolhuter

March 2007

Copyright © 2007 University of Stellenbosch
All rights reserved.



Declaration

I, the undersigned, hereby declare that the work contained in this thesis is my own original work and that I have not previously in its entirety or in part submitted it at any university for a degree.

Signature:

D.W. Morrison

Date:



Abstract

Using ad hoc wireless networks to enable intelligent transport systems: The design and analysis of the TH(O)RP routing protocol

D.W. Morrison

*Department of Electronic Engineering
University of Stellenbosch
Private Bag X1, 7602 Matieland, South Africa*

Thesis: MScEng (Electronic Engineering)

March 2007

With the rapid advancement of communication technologies and broadband communication, an era is starting to emerge where everything and everyone is always connected, regardless of geography. No other technology has made this more possible than over-the-air data communications technologies such as Wi-Fi, WiMAX and cellular technologies.

With the possibility of connecting more devices to a common communications network, more and more applications become available and necessary. One such application is a concept designed to manage a different type of network; the traffic networks of large metropolitan areas. These networks carry more traffic with each passing year and the need to manage them efficiently has become essential. A system to manage traffic networks is an intelligent transport system (ITS), which integrates all methods of transportation into a single manageable resource. Information about the current status of the traffic network can be relayed to road users, allowing them to make informed decisions about alternative routes, or to emergency personnel to inform them of accidents that occurred on the traffic networks. In order to implement an ITS, a communication network is required.

This thesis investigates different communication technologies, discussing their merits and shortcomings in an ITS implementation. A suitable technology is selected and a communications system is conceptualised. The communications system is an ad hoc wireless network and a routing protocol used to manage the network, is designed and tested through simulation.

The TH(O)RP routing protocol was developed with a focus on scalability, stability and low latency in an ad hoc network. TH(O)RP was designed to operate in an ITS environment, where traffic intersection controllers (TIC) are monitored from a central entity, with optimal routes between the central entity and the TICs, that can be automatically configured and repaired.

Opsomming

Die gebruik van ad hoc draadlose netwerke om intelligente vervoer stelsels moontlik te maak: Die ontwerp en analise van die TH(O)RP netwerkprotokol

(“Using ad hoc wireless networks to enable intelligent transport systems: The design and analysis of the TH(O)RP routing protocol”)

D.W. Morrison

Departement Elektroniese Ingenieurswese

Universiteit van Stellenbosch

Privaatsak X1, 7602 Matieland, Suid Afrika

Tesis: MScIng (Elektroniese Ingenieurswese)

Maart 2007

Met die spoedige groei van kommunikasietegnologieë wat tans plaasvind, begin ons in 'n tydperk beweeg waar dit moontlik raak om alles en almal te verbind, maak nie saak waar jy jouself bevind nie. Geen ander tegnologie het dit meer moontlik gemaak as draadlose kommunikasie tegnologieë, of meer spesifiek Wi-Fi, WiMAX en sellulêre tegnologieë.

Met die moontlikheid dat alles verbind kan word, word al hoe meer toepassings moontlik en noodsaaklik. Een so 'n toepassing word ontwikkel met die doel om 'n ander netwerk te bestuur, die intelligente vervoer stelsel (IVS), wat alle aspekte van 'n vervoernetwerk saam groepeer en as 'n eenheid bestuur. Inligting oor die huidige toestand van die vervoernetwerk kan aan pendelaars bekend gemaak word sodat hulle ingeligte besluite kan neem oor alternatiewe roetes, asook noodpersoneel wat die inligting kan gebruik om vinniger op ongelukstonele betrokke te raak. 'n Stelsel soos hierdie het wel 'n kommunikasienetwerk nodig om te funksioneer.

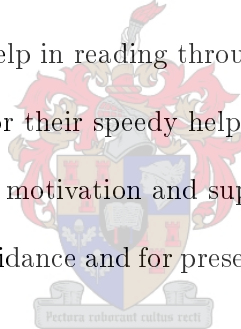
Hierdie tesis ondersoek verskillende kommunikasietegnologieë en bespreek hul voordele en nadele met die fokus op 'n toepassing in 'n IVS omgewing. Vanuit hierdie tegnologieë is 'n toepaslike kandidaat geselekteer en 'n gevolglike kommunikasiestelsel is ontwikkel en getoets. Die kommunikasiestelsel is 'n ad hoc draadlose netwerk en die protokol wat die roetes deur die stelsel onderhou, word ontwerp en deur simulasies getoets.

Die TH(O)RP netwerkprotokol was ontwikkel met die oog op skalering, stabiliteit en laë wagtye in 'n ad hoc draadlose netwerk. TH(O)RP was ontwerp vir gebruik in 'n IVS omgewing, waar optimale roetes tussen 'n sentrale entiteit en die nodusse van 'n netwerk van belang is, en waar die roetes outomaties herstel moet word in die geval van netwerk-onderbrekings.

Acknowledgements

I would like to express my sincere gratitude to the following people and organisations who have contributed to making this work possible:

- The Traffic Directorate of the City of Cape Town for presenting us with this topic.
- My supervisor, Dr. Riaan Wolhuter, for his guidance and support, and for encouraging me to finish this thesis.
- My girlfriend Mariecha for her love, support and for motivating me to finish this thesis.
- My parents for their love, emotional- and financial support through the years.
- My brother Campbell, for his help in reading through an early draft of this thesis.
- The OMNET++ mailing list, for their speedy help and response to my questions.
- My family and friends, for their motivation and support during the course of this project.
- My Heavenly Farther, for His guidance and for presenting me with opportunities and giving me the talents to pursue them.



Contents

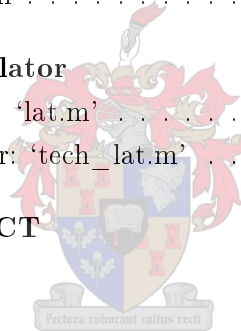
Declaration	ii
Abstract	iii
Opsomming	iv
Acknowledgements	v
Contents	vi
List of Figures	xi
List of Tables	xv
Nomenclature	xvi
1 Introduction	1
1.1 The need for intelligent transport systems	1
1.2 Using ad hoc wireless networks to facilitate ITS	1
1.3 Project objectives	2
1.4 Contributions	3
1.5 Thesis overview	3
I Feasibility study on a metropolitan wide networked communications solution for traffic intersection controllers	5
2 Communication alternatives for an ITS system	6
2.1 Feasibility study overview	6
2.1.1 Goal of this study	6
2.1.2 Modus operandi	6
2.1.3 Telecommunication network specification	7
2.1.4 Communication technologies	7
2.1.5 Summary	7
2.2 Technology investigation	7
2.2.1 Overview	7
2.2.2 Wired technologies	8
2.2.3 Wireless technologies	8

2.2.4	Cellular technologies	10
2.2.5	Hardware bandwidths	12
2.2.6	Comparing technologies	13
2.2.7	Estimated technology latencies	13
2.2.8	Summary	17
2.3	Proposed ITS communication network	17
2.3.1	Overview	17
2.3.2	Additional network functionality	17
2.3.3	Conceptual network design	19
2.3.4	Conclusion	21
2.4	Summary	22
II Ad hoc wireless networking protocol development and evaluation		23
3	Theoretical background	24
3.1	Introduction	24
3.2	Ad hoc wireless networks: a routing problem	24
3.3	Multi-hop routing protocols	25
3.3.1	Proactive vs. reactive vs. hybrid approach	25
3.3.2	Flat routing	26
3.3.3	Hierarchical routing	27
3.4	Conclusion	28
4	The tree hierarchy (open) routing protocol (TH(O)RP)	29
4.1	Introduction	29
4.2	Design goals	29
4.3	Why a tree?	30
4.4	TH(O)RP operation	31
4.5	Primary components	33
4.5.1	Neighbour discovery and link sensing	34
4.5.2	Parent selection	35
4.5.3	Route discovery and maintenance	36
4.5.4	Message forwarding	36
4.5.5	Unknown message handling	37
4.6	Secondary components	38
4.6.1	Link hysteresis	38
4.6.2	Link metric	39
4.6.3	Message delivery and indirect acknowledgements	41
4.7	Packet formats	43
4.8	Modular protocol design	44
4.8.1	Socket parser	45
4.8.2	Packet parser	45
4.8.3	Information repositories	45
4.8.4	Scheduler	46

4.8.5	Plug-in interface	46
4.9	Protocol development cycle	47
4.10	Future functionality	47
4.10.1	Multiple interface declaration	47
4.10.2	Internet connectivity via host and network association	48
4.10.3	Security	48
4.10.4	Better hierarchy structure via node willingness	49
4.10.5	Quality of Service	49
4.11	Conclusion	49
5	Networking latency	50
5.1	Terms and Definitions	50
5.1.1	Bandwidth	50
5.1.2	Latency	50
5.2	Sources of latency	51
5.3	Assumptions for latency models	51
5.4	Modeling latency with queueing theory	51
5.4.1	The basic latency model	51
5.4.2	One hop latency models	53
5.4.3	Multiple hop latency models	53
5.4.4	Latency models with retransmissions due to noise	54
5.5	Developing latency models based on TH(O)RP	54
5.5.1	Communications protocol breakdown	54
5.5.2	Routing protocol	55
5.5.3	Packet nesting	57
5.5.4	Transport- and MAC protocol overhead	58
5.6	Latency models	58
6	TH(O)RP simulation setup	61
6.1	Introduction	61
6.1.1	Simulation overview	61
6.1.2	OMNET++ and the mobility framework	61
6.1.3	Modus Operandi	62
6.1.4	Summary	62
6.2	Model overview and simulation environment	62
6.2.1	Overview	62
6.2.2	Node composition	63
6.2.3	Layers of the nodes	64
6.2.4	Network messages	66
6.2.5	Node status colour coding and bubble messages	66
6.2.6	Graphical tree hierarchy representation	66
6.2.7	Simulation statistics and results collection	70
6.2.8	Route table storage requirements	71
6.2.9	Simulation setup	72

6.2.10	Summary	72
6.3	Simulation scenarios	73
6.3.1	Overview	73
6.3.2	Scenario network layout	74
6.3.3	Summary	75
7	TH(O)RP simulation results	76
7.1	Initial simulation results	76
7.1.1	Overview	76
7.1.2	Comparison against latency model results	76
7.1.3	Optimising TH(O)RP timing- and retry parameters	77
7.1.4	Improved acknowledgement and retransmission cooperation	85
7.1.5	The effect of communications data rate on throughput and latency	86
7.1.6	The effect of multiple hops on TH(O)RP	87
7.1.7	The responsiveness of TH(O)RP to route failures	89
7.1.8	Route metric evaluation	94
7.1.9	Localised property of TH(O)RP	95
7.1.10	Testing TH(O)RP scaling	96
7.1.11	Effect of startup time	98
7.1.12	Summary	98
7.2	Introduction of the fast recovery methods	99
7.2.1	Overview	99
7.2.2	Method 1: Variable parent selection interval	99
7.2.3	Method 2: Assisted high speed propagation path (AHSP)	99
7.2.4	Method 3: Adoption notification	100
7.2.5	Method 4: Route error notification	100
7.2.6	Method 5: Cumulative route request (CRREQ)	100
7.2.7	Summary	101
7.3	Revisited simulation results	101
7.3.1	Overview	101
7.3.2	Responsiveness tests revisited	101
7.3.3	Overall impact of fast recovery on TH(O)RP performance	102
7.3.4	Summary	103
7.4	Conclusion	103
8	C implementation of TH(O)RP	105
8.1	Introduction	105
8.2	Application composition	105
8.3	Application output	105
8.3.1	Information repository output	106
8.3.2	TH(O)RP packet output	106
8.4	Execution and input options	108
8.5	Virtual network: thrp_switch	108
8.6	Future work	110

8.7	Conclusion	110
9	Conclusion	111
9.1	Overview of work	111
9.2	Further work	113
9.3	Summary	115
	Bibliography	116
	Appendices	119
A	Packet formats and headers	120
A.1	TH(O)RP packet and message formats	120
A.2	Miscellaneous routing, transport and MAC layer packet and frame formats	121
B	TH(O)RP message definitions in OMNET++ simulation	124
C	Configuration files for OMNET++ simulations	132
C.1	Layout file: layout.ini	132
C.2	Parameter file: parameters.ini	140
D	Matlab code for latency calculator	142
D.1	Individual latency calculator: 'lat.m'	142
D.2	Technology latency calculator: 'tech_lat.m'	147
E	Proposal presented to the CoCT	150



List of Figures

2.1	Common wireless network applications	10
	(a) Common application of Wi-Fi [9]	10
	(b) Common application of WiMAX [10]	10
2.2	Evolution of 3G[15]	10
2.3	Comparing the old and suggested network topologies	20
	(a) Original network topology [25]	20
	(b) Suggested network topology	20
2.4	Abstraction of network architecture	21
2.5	Abstraction of a multi-radio configuration	21
4.1	An example network and the revealed tree hierarchy	31
	(a) Example network	31
	(b) Revealed tree hierarchy	31
4.2	Illustration of route discovery with the high-speed propagation path (HSPP)	32
4.3	Illustration of route discovery between two nodes	33
4.4	Illustration of reduced flooding	33
	(a) Network wide flooding: all nodes participating	33
	(b) Network wide flooding: parent nodes participating	33
4.5	Calculating link state during link sensing [46]	35
4.6	Illustration parent selection process	35
4.7	Illustration of default forwarding strategy	37
4.8	Illustration of link hysteresis	39
4.9	Comparison of two acknowledgement techniques	42
	(a) Per hop acknowledgements	42
	(b) End-to-end acknowledgements	42
4.10	Indirect acknowledgements	43
4.11	The Generic TH(O)RP packet	44
4.12	Modular design of TH(O)RP	44
4.13	Example of a security plug-in	47
4.14	The protocol development cycle	48
5.1	Basic latency model	52
5.2	One hop queueing models	53
	(a) 1:1 model	53
	(b) 1:N model	53

(c)	N:1 model	53
5.3	Multi-hop queueing models	54
(a)	Simple multi-hop (1:1:1) model	54
(b)	Complex multi-hop model (N:N:1)	54
(c)	Complex multi-hop model (1:N:N)	54
5.4	The OSI reference model [45]	55
5.5	The effects of the forwarding jitter	57
(a)	Forwarding jitter effect on message generation	57
(b)	Forwarding jitter effect on forwarded messages	57
5.6	Packet nesting	58
6.1	Example simulation setup with 10 outstations and central instation	62
6.2	The different layers of the network nodes	63
6.3	Network layer output during simulation	64
(a)	Output of the instation.	64
(b)	Output of the an outstation.	64
6.4	Application layer output of instation.	65
6.5	Illustration of TH(O)RP packet and contents	67
(a)	TH(O)RP packet with multiple message	67
(b)	Encapsulated TRAFX packet	67
6.6	Simulation visual aids	68
(a)	Colour codes: off, active but orphaned, active with parent	68
(b)	Bubble messages	68
6.7	Text output of create_dot_graph script.	69
6.8	Graphical output graphs created by ‘create_dot_graph’ script	69
(a)	All connections	69
(b)	Parent connections only	69
6.9	Collected statistics for example simulation for the instation	70
6.10	Collected statistics for example simulation for outstations	71
(a)	Oustation one	71
(b)	Outstation two	71
6.11	Latency comparison simulation scenario	74
6.12	20 Node scenario	74
6.13	The 36 and 100 node scenarios	75
(a)	36 Node scenario	75
(b)	100 Node scenario	75
7.1	HELLO message optimisation effect on latency	79
7.2	Effect of maximum forwarding jitter on latency	80
7.3	Effect of retry timeout on latency	82
7.4	Effect of poor link qualities and retransmissions on latency	82
7.5	Effect of the number of retransmissions on latency	83
7.6	Effect of the number of retransmissions on latency, a second test	84
7.7	Different acknowledgement scenarios	85
(a)	Successful implied ACK reception	85

(b)	Implied ACK lost, retransmissions ignored	85
(c)	Implied ACK lost, send direct ACK on retransmissions	85
7.8	Histogram of retransmission distribution	86
7.9	Improved histogram of retransmission distribution	86
7.10	Effect of data rate on latency	87
7.11	Effect of multiple hops on latency (0% lossy channels)	89
7.12	Effect of multiple hops on latency (10% lossy channels)	89
7.13	Graphical representation of matrix scenario tree hierarchy with node 43 highlighted	90
7.14	Example of using bubble messages to track message progress	91
7.15	Tree hierarchy graphical representation of recovery tests	92
(a)	Test 1	92
(b)	Test 2	92
(c)	Test 3	92
(d)	Test 4	92
(e)	Test 5	92
7.16	Illustration of effect of test 5 on nodes.	93
7.17	New recovered route after node failures in test 5.	93
7.18	Illustration of an alternative route established with the route metric	94
(a)	Alternative route around poor network area	94
(b)	Tree hierarchy of alternative route	94
7.19	Localised property of TH(O)RP	95
7.20	Network layer output of nodes at different levels of tree hierarchy	97
(a)	Instation output	97
(b)	Middle outstation output	97
(c)	Bottom outstation output	97
8.1	Target network hierarchy for illustration of TH(O)RP output	106
8.2	Network layer output during execution of TH(O)RP	107
(a)	Output of node 10.0.0.1	107
(b)	Output of node 10.0.0.2	107
(c)	Output of node 10.0.0.4	107
8.3	TH(O)RP packets and messages	107
(a)	HELLO message	107
(b)	RREQ message	107
(c)	RREP message	107
8.4	Usage summary of TH(O)RP	109
8.5	thrp_switch commands used to create network topology for Figure 8.1	109
A.1	TH(O)RP packet- and message format	120
A.2	The HELLO and RREQ messages	120
(a)	HELLO message format	120
(b)	Route request message format	120
A.3	Data, ACK and TRAFX messages	121
(a)	Data message format	121
(b)	ACK message format	121

(c) TRAFX message format	121
A.4 IPv4 packet format	121
A.5 UDP packet format	122
A.6 TCP packet format	122
A.7 Ethernet (802.3) packet format	122
A.8 Wi-fi (802.11) packet format	122
A.9 WiMAX (802.16) packet format: Generic and bandwidth requested	123
A.10 GSM packet format	123



List of Tables

2.1	Task groups and family specifications [44]	9
2.2	Device bandwidths [12]	12
2.3	Technology comparison	13
2.4	Resulting latencies for wired technologies (jitter_max = 0.125s)	14
2.5	Resulting latencies for wireless technologies (jitter_max = 0.125s)	15
2.6	Resulting latencies cellular technologies (jitter_max = 0.125s)	16
4.1	Comparison of two types of acknowledgements	42
5.1	TH(O)RP message and header sizes	56
5.2	Generation rates	57
5.3	Transport- and MAC protocol header sizes	58
5.4	Latency model parameters for TH(O)RP routing protocol (N=2)	60
5.5	Resulting data sizes	60
6.1	Simulation scenarios and objectives	73
7.1	Comparisons of latencies for Wi-Fi (model vs. simulations)	77
7.2	HELLO message optimisation summary (default marked with *)	78
7.3	Maximum jitter optimisation summary (default marked with *)	79
7.4	Effect of optimised hash algorithm on duplicate set storage	81
7.5	Effect of retry time on success rate (default marked *)	81
7.6	Effect of number of retransmissions on success rate (default marked with *)	83
7.7	Effect of number of retransmissions on success rate, a second test	84
7.8	Effect of data rate on success rate	87
7.9	Effect of multiple hops of throughput success rate	88
7.10	Route recovery times	92
7.11	Localised property of TH(O)RP	95
7.12	Control traffic rates at different levels in tree hierarchy	98
7.13	Route table storage requirements	98
7.14	Synchronised versus unsynchronised startup times for nodes	98
7.15	Route recovery times revisited	102
7.16	Effect of fast recovery methods on TH(O)RP	103
8.1	Command line input parameters for TH(O)RP	108

Nomenclature

Acronyms:

<i>16QAM</i>	16-State quadrature amplitude modulation
<i>ACK</i>	Acknowledgement message
<i>AHSPP</i>	Assisted high-speed propagation path
<i>AODV</i>	Ad hoc on-demand distance vector
<i>bit (b)</i>	A single bit, 0 or 1, of data
<i>byte (B)</i>	Eight bits
<i>BPL</i>	Broadband over power lines
<i>C</i>	The C software programming language
<i>C++</i>	A Object oriented extension of the C programming language
<i>CCTV</i>	Closed circuit television
<i>CDMA</i>	Code division multiple access
<i>CDMA2000</i>	Code division multiple access 2000
<i>CoCT</i>	City of Cape Town
<i>CRREQ</i>	Cumulative route request message
<i>CS-4</i>	Coding scheme 4 for GPRS
<i>CTS</i>	Clear to send
<i>DoS</i>	Denial of service
<i>DSDV</i>	Destination-sequence distance vector
<i>DSL</i>	Digital subscriber line (as in xDSL)
<i>EDGE</i>	Enhanced data rates for GSM evolution
<i>ETT</i>	Expected transmission time
<i>ETX</i>	Expected transmission cost
<i>EV-DO</i>	1 x Evolution-data optimised
<i>GPL</i>	Gnu general public licence
<i>GPRS</i>	General packet radio service
<i>GSM</i>	Global system for mobile communications
<i>HELLO</i>	HELLO message
<i>HNA</i>	Host and network association
<i>HS-DSCH</i>	High-speed downlink shared channel
<i>HSDPA</i>	High-speed downlink packet access
<i>HSPP</i>	High-speed propagation path
<i>IEEE</i>	Institute of electrical and electronics engineers
<i>Instation</i>	Central controller (ROOT of the tree hierarchy of TH(O)RP)
<i>IP</i>	Internet protocol (IPv4 and IPv6)

<i>ITS</i>	Intelligent transport system
<i>LAN</i>	Local area network
<i>LOS</i>	Line of sight (0-48km)
<i>MAC</i>	Medium access control
<i>MAN</i>	Metropolitan area network
<i>MARCH</i>	Multiple access with reduced handshake
<i>MID</i>	Multiple interface declaration
<i>MPR</i>	Multi-point relay
<i>Msg</i>	Message
<i>NIC</i>	Network interface controller
<i>OC</i>	Optical carrier
<i>OFDM</i>	Orthogonal frequency division multiplexing
<i>OLSR</i>	Optimised link state routing
<i>OSI</i>	Open systems interconnection
<i>Outstation</i>	Traffic intersection controller (branch or leaf of the tree hierarchy of TH(O)RP)
<i>PHY</i>	Physical layer device/protocol
<i>PLC</i>	Power line communication
<i>QoS</i>	Quality of service
<i>QPSK</i>	Quadrature phase shift keying
<i>ROOT</i>	The top of the tree hierarchy of TH(O)RP
<i>RREQ</i>	Route request message
<i>RREP</i>	Route reply message
<i>RERR</i>	Route error message
<i>RTS</i>	Request to send
<i>RTT</i>	Round-trip-time
<i>RX</i>	Receive/Receiver/Reception
<i>SCATS</i>	Sydney coordinated adaptive traffic system
<i>SCOOT</i>	Split cycle and offset optimisation technique
<i>SP</i>	Service provider
<i>SSL</i>	Secure socket layer
<i>STL</i>	Secure transport layer
<i>TBRPF</i>	Topology dissemination based on reverse-path forwarding
<i>TDM</i>	Time division multiplexing
<i>TDMA</i>	Time division multiple access
<i>TH(O)RP</i>	Tree hierarchy (open) routing protocol
<i>TIC</i>	Traffic intersection controller(s) or signalised intersection controller(s)
<i>TCP</i>	Transmission control protocol
<i>TTL</i>	Time to live
<i>TX</i>	Transmit/Transmitter/Transmission
<i>UDP</i>	User datagram protocol
<i>VIP</i>	Video IP
<i>VMS</i>	Variable message sign
<i>VoIP</i>	Voice over IP
<i>VPN</i>	Virtual private network

<i>WCDMA</i>	Wideband code division multiple access
<i>WCETT</i>	Weighted cumulative ETT
<i>WCETX</i>	Weighted cumulative ETX
<i>WLAN</i>	Wireless local area network
<i>Wi-Fi</i>	Wireless fidelity (IEEE 802.11 wireless networking standard)
<i>WiMAX</i>	Worldwide interoperability for microwave access (IEEE 802.16d wireless broadband standard)
<i>UDP</i>	User datagram protocol
<i>UTC</i>	Urban traffic control
<i>UMTS</i>	Universal mobile telecommunications system
<i>ZRP</i>	Zone routing protocol

Operators:

Σ	Sum
\cdot	Multiplication

Constants:

$c = 3 \cdot 10^8$	Speed of light
--------------------	----------------

Variables:

λ	Arrival rate
μ	Service rate
ρ	Traffic intensity
p_{err}	Probability of transmission error
p_f	Forward loss probability
p_r	Reverse loss probability
$p_{success}$	Probability of successful packet transmission
$s(k)$	Probability that packet is successfully delivered after k attempts
t_{bit}	Time required to transmit a single bit of data
t_{data}	Time required to transmit D bits
$t_{control}$	Time required to transmit control traffic
t_{packet}	Time required to transmit a packet, including control overhead
t_{retry}	Time required to retransmit corrupted or lost packets
t_{total}	Time required to transmit a packet in the presence of noise
T_{wait}	Latency of a communication network
T_{wait_jitter}	Latency of a communication network including wait times



Chapter 1

Introduction

1.1 The need for intelligent transport systems

Traffic networks have become increasingly complex and congested as the demand and use of these networks increased. This puts strain on the entire system and causes chronic congestion if not actively managed. An intelligent transport system (ITS) allows transport networks to be monitored and managed, increasing the overall efficiency of usage of transport networks. ITS includes monitoring traffic flow via closed circuit television (CCTV) systems or other methods, for example, the Sydney coordinated adaptive traffic system (SCATS) [17]. SCATS uses traffic cameras and induction loops within the road paving to count vehicles at traffic intersections, and adapts the timing of intersection controllers through a centralised computer. ITS allows information regarding the state of routes to be relayed to commuters, enabling them to make informed decisions about using alternative routes. It also promotes public transport by managing routes reserved for public transport, or public transportation resources can be deployed to areas where it is most needed. Emergency personnel can use information gathered by ITS to promptly react to developing emergency situations. ITS can also be employed, to promote increased cooperation between various emergency departments.

The City of Cape Town (CoCT) is in a predicament with its traffic networks. The present networks are chronically congested during the morning and afternoon rush-hour, and due to the location of the city between Table Mountain and the ocean, it is unable to expand its traffic networks. This stresses the need for the traffic networks to be managed more efficiently. At present, the CoCT has a partially deployed ITS system that uses the split cycle and offset optimisation technique (SCOOT). This is an area traffic control system, that similar to the SCATS system, uses a central computer to monitor TICs. This system needs to be expanded considerably, but the communication infrastructure required to deploy the ITS system is lacking. A solution is desired that would provide sufficient communication infrastructure to allow the deployment of an ITS system.

1.2 Using ad hoc wireless networks to facilitate ITS

Most of the major cities of the world have well established communication infrastructures and are able to employ an ITS system on top of these infrastructures. In other regions of the world the communication infrastructures can be poorly developed, which is particularly true in third world

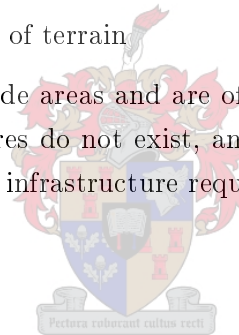
countries and in remote areas. In these scenarios, traditional wired infrastructures most often do not exist or have too little capacity and rolling out sufficient wired infrastructures would either take too long to complete, or carry too high a cost. In these situations wireless communication technologies become an attractive option.

Ad hoc wireless networks are defined as the category of wireless networks that utilise multi-hop radio relaying and are capable of operating without the support of any fixed infrastructure [39]. This allows for rapid deployment of these networks over vast distances. They are also extremely robust, as nodes in such a network usually have multiple routes between any two points in the network, and in the event of a network failure it can use this redundancy in order to continue operating. In addition to this, these networks are usually self-configuring and self-healing, and they require a minimal amount of maintenance to keep operating. Due to their wireless network interfaces, these networks can communicate over water, buildings and wide areas. Some of the advantages offered by ad hoc wireless networks are:

- robust, with built in redundancy
- easily expandable
- automatic configuration and recovery
- can cover wide areas, regardless of terrain

Because ITS systems usually span wide areas and are often required to operate in areas where previous communication infrastructures do not exist, an ad hoc wireless network could be employed to provide the communication infrastructure required by the ITS system.

1.3 Project objectives



At the outset of this project, a number of objectives were identified:

- A study of communication technologies must be conducted, investigating their advantages and disadvantages for use in a communication network used to deploy an ITS system. A communication solution should also be proposed.
- An ad hoc wireless routing protocol that could be used to manage the communication network proposed in the study, should be designed. Design goals of the protocol should include automatic configuration and recovery, scalability, stability and low latency in large, dynamic networks.
- The latencies of different communication technologies using the routing protocol designed, should be theoretically modeled and estimated.
- The routing protocol must be thoroughly analysed through simulation. During simulation the protocol must also be improved and optimised.

It is important to note that the solution proposed in the project, as well as the subsequently developed routing protocol, are not meant to be fully featured. They are meant to serve as proof of concept for the use of ad hoc wireless networks, to manage large communication networks that could facilitate ITS systems. The development and analysis of the ad hoc routing protocol is the focal point of this project.

1.4 Contributions

In accomplishing the project objectives, a number of contributions were achieved.

- A study of alternative solutions for communication infrastructures to implement ITS systems could be presented to the CoCT.
- The development of an ad hoc wireless routing protocol, that is well suited to manage communication networks used to implement ITS systems.
- A theoretical latency model to deterministically predict the performance of such networks.
- A simulation tool to use in conjunction with the theoretical model, for quick and sufficiently accurate modeling of a very wide range of network topologies.
- The stimulation of further research and development of ad hoc wireless routing protocols and open standards, by releasing any software developed under the Gnu general public licence (GPL).

1.5 Thesis overview

This thesis is divided into two distinct parts.

In Part I, a feasibility study is presented. This study was conducted upon request from the CoCT, to investigate alternative communication solutions that could be used to deploy an ITS.

The study is discussed in Chapter 2. The merits and shortcomings of different communication technologies for use in an ITS implementation, are investigated. It also provides an estimation of the latencies of the different technologies in a multi-hop network environment. Additional services that could be employed over a network used to facilitate an ITS system, is discussed. Finally, a communication network is conceptualised.

Part II contains the development of a multi-hop wireless routing protocol, that would be used to manage the communication network conceptualised in Part I.

A study of different ad hoc routing protocols is conducted in Chapter 3. The different categories of ad hoc protocols are discussed, along with examples of some of the most popular protocols. The advantages and disadvantages of these protocols are identified.

Chapter 4 provides a detailed description of the multi-hop routing protocol developed in this thesis. It describes the operation of the protocol, and discusses future improvements that could be made to the protocol.

In Chapter 5 a theoretical study of networking latency is conducted. Latency models are developed that could approximate the operation of the protocol developed in the previous chapter, and the models are used to estimate the latencies that the protocol would experience, when using different communication technologies. The results from the latency models were included in Section 2.2.7 of the feasibility study, as the results could verify the applicability of each technology for use in the communication network, envisioned for the ITS system of the CoCT.

A detailed analysis of the multi-hop protocol is conducted in Chapters 6 and 7. In Chapter 6 a simulation of the protocol, which was developed in the C++ programming language, is discussed and test cases are set up. In Chapter 7 the protocol is extensively tested and optimised.

Additional methods are also developed to improve the protocol. The results of the simulations are discussed, along with the strengths and weaknesses of the protocol.

A lightweight software implementation of the multi-hop protocol, developed in the C programming language, is discussed in Chapter 8. The execution of this implementation and future improvements are discussed.

Chapter 9 concludes the document and provides an overview of the the statements and conclusions made in this document.



Part I

Feasibility study on a metropolitan wide networked communications solution for traffic intersection controllers



Chapter 2

Communication alternatives for an ITS system

2.1 Feasibility study overview

2.1.1 Goal of this study

The City of Cape Town (CoCT) uses a traditional circuit switched communication network to communicate with the signalised traffic intersection controllers (TIC) throughout the city. Presently, the communication network is supplied by Telkom by means of copper telephone lines and utilised by V23 modems [24]. Telkom has recently upgraded their analogue networks to digital networks and no longer support the analogue services supplied to the CoCT. With the move to digital systems, came a significant increase in cost.

The goal of this study is to find a communication network alternative to the options presently available from Telkom. This study will look at current communication technologies and verify whether or not they can provide the CoCT with an alternative communication network that can meet their telecommunication requirements, and if so, what other services they will be able to facilitate. From these technologies a suitable solution can be selected to provide communication with the TICs in the CoCT.

This study was originally intended to be presented to people that have a limited technical background in telecommunication networks and technologies. It was also developed as a separate document which could be presented to the CoCT. Due to this, there might be some duplication of content in part two of this thesis.

2.1.2 Modus operandi

This study proceeds by specifying the telecommunication requirements of the CoCT. This is followed by a brief overview of the different communication technologies that are investigated. Then the capabilities and limitations of the hardware is assessed, stating whether or not they are able to meet the needs of the desired communication network. Some of the limitations will include line of sight (LOS) requirements, maximum usable distance, effective throughput and whether a third party service provider is involved. After this, the additional services that the network would be able to accommodate, is discussed, namely VMS, CCTV and Internet connectivity. Finally a solution for the communication system is proposed.

2.1.3 Telecommunication network specification

The CoCT is looking for the most appropriate telecommunication backbone to facilitate their current communication.

The new telecommunication network must be reliable, easily expandable and robust. It must, therefore, be able to continue operating in an environment where nodes of the network could go off-line.

The telecommunication network must have a low latency, as an one second maximum latency is required to accommodate the SCOOT traffic monitoring software being used on the TICs. Thus the network needs to allow communication to take place between an instation (central controller) and each of the outstations (TIC) every second. This is the maximum latency, so a latency lower than one second is desirable.

The telecommunication network will also need to be time verifiable, with time stamped data, within tight parameters. This will require a time server to be included in the communication network. Geocoded information about the TICs will also need to be available, so that individual nodes and their location in the CoCT can be identified.

The new telecommunication network will need to be able to accommodate the TRAFX communication protocol used by the urban traffic control (UTC) messaging system [1], as defined in [25]. This protocol enables the TICs to monitor traffic and communicate with the SCOOT instation. It will also enable the uploading of data to the TICs, for example uploading of traffic flow algorithms to outstations.

2.1.4 Communication technologies

The communication technologies that are investigated, fall under three main classes, namely wired, wireless and cellular technologies. These are listed below:

Class	Name
Wired	Fibre optics, Broadband over power lines (BPL)
Wireless	Wi-Fi, WiMAX
Cellular	GPRS, EDGE, 3G, HSDPA

2.1.5 Summary

A study of the hardware capabilities is conducted with a specific focus on latency. A solution is proposed, but the decision of which solution to select and how to develop it will remain with the CoCT. The feasibility study only aims to give the CoCT the information required to make an informed decision regarding technologies available today, so they are able to select the most appropriate solution for their communication requirements.

2.2 Technology investigation

2.2.1 Overview

In this section a brief overview of the different communication technologies is supplied, to provide the CoCT with some background information of these technologies. As stated before, the technologies are divided into three categories, namely wired, wireless and cellular.

2.2.2 Wired technologies

Fibre optics

An optical fibre is a thin, transparent fibre, usually made from glass or plastic, for transmitting light. It operates through a process of total internal reflection, which allows light to be transmitted along the axis of the fibre with little or no loss of light energy [14]. These fibres are also named optical carriers (OC). Information can then be transmitted via the fibre as pulses of light. Fibres can be grouped inside a single cable and can achieve high bandwidths, up to 40Gb/s (5GB/s).

Deployment of a fibre optic network is rather expensive due to the tunnelling and digging needed to install the cables underground. Expansion of such a network is also difficult due to the nature of any wired infrastructure. A fibre optic network can however provide a high bandwidth backbone for another network to connect to.

Broadband over power lines

Power line communication (PLC) [16] is a method through which power distribution wires is used for simultaneous distribution of data. A service to provide broadband data access over PLC is broadband over power lines (BPL) which has data rates ranging from 256kb/s to 2.7Mb/s. This could be used to provide broadband Internet access through ordinary power lines.

BPL systems can be combined with other wired or wireless infrastructures, to solve the last mile problem for providing Internet access to many homes and devices.

This is advantageous to the CoCT, because all of the TICs are already connected to power lines, so in effect a wired infrastructure already exists that can be utilised to employ an ITS system in the CoCT. Goal Technology Solutions [8] currently provides a BPL service in South Africa.

2.2.3 Wireless technologies

Wi-fi (802.11)

Wi-Fi is a term that describes the underlying technology of wireless local area networks (WLAN) based on the IEEE 802.11 specifications, which transmits in the 2.4GHz and 5.8GHz frequency bands.

Wi-Fi was intended to be used for mobile computing devices in LANs, but has been adopted for many other applications, including Internet access, gaming, and basic connectivity of consumer electronics such as televisions and DVD players. More standards are being developed, that will allow Wi-Fi to be used by cars in highways in support of ITS [19]. Wi-Fi was developed for internal networks, but with the right antennas it has been expanded to providing connectivity in external networks. Speeds of 11Mb/s to 54Mb/s can be achieved. There are many different flavours of Wi-Fi, as summarised in Table 2.1.

Deployment of wireless networks are fundamentally easier than wired networks, as no tunnelling is required for cables. All that is required is to add a Wi-Fi device to the desired location, within range of the existing wireless network, and to configure the network to allow the device connectivity. Due to this, a wireless network roll-out is much easier and cheaper than a wired network and is easily expanded.

Table 2.1: *Task groups and family specifications [44]*

Task Group	Task	Maximum Transfer Rate/Frequencies
802.11	Wireless LAN PHY and MAC specifications (infrared and 2.4 GHz radio)	
802.11a	Wireless LAN PHY and MAC specifications for 5.8GHz radio band	54Mbps/5.8GHz
802.11b	Higher-Speed (5.5 Mbps and 11 Mbps) WLAN PHY and MAC specifications for 2.4GHz	11Mbps/2.4GHz
802.11c	Bridge operation with IEEE 802.11 MACs (incorporated into 802.1d)	
802.11d	Extensions to 802.11 for operation in additional regulatory domains	
802.11e	802.11 MAC Quality of Service (QoS) for advanced applications	
802.11f	Multi-vendor access point interoperability across distribution systems: IAPP	
802.11g	Higher-Rate extensions in the 2.4 GHz radio band	54Mbps/2.4GHz
802.11h	Enhancements for dynamic channel selection and transmit power control	
802.11i	Enhancements for security and authentication	

Due to the open nature of a wireless network, steps to secure the network must be taken. Unless adequately protected, a Wi-Fi network can be susceptible to access by unauthorised users, who can use the access for personal, illegal or malicious purposes. Any entity that has a wireless LAN should use security safeguards such as the wired equivalent privacy (WEP) encryption standard, the more recent Wi-Fi protected access (WPA), Internet protocol security (IPsec), or a virtual private network (VPN) [13].

A wireless network can be used as a stand-alone entity, or can be used in conjunction with existing wired infrastructures to expand the existing network.

WiMAX (802.16d)

Commonly referred to as WiMAX, also as WirelessMANTM or the Air Interface Standard, IEEE 802.16d is a specification for fixed broadband wireless metropolitan access networks (MANs) that use a point-to-multi-point architecture. The standard defines the use of bandwidth between the licensed 10-66GHz and the unlicensed 2-11GHz frequency ranges. It also defines a MAC layer that supports multiple physical layer specifications customised for the frequency band of use and their associated regulations. 802.16d supports high bit rates in both uploading to and downloading from a base station up to a distance of 48 km to handle such services as VoIP, IP connectivity and TDM voice and data [3].

WiMAX is designated as the metropolitan area network (MAN) technology that can connect IEEE 802.11 (Wi-Fi) hot spots with each other and to other parts of the Internet and provide a wireless alternative to cable and DSL for last mile broadband access.

In theory, WiMAX can provide up to 48km of linear service area range and allows connectivity between users without line of sight (LOS). The technology has also claimed to provide shared data rates of up to 70Mb/s. Real world tests however, show the maximum operating distance to be around 5-8km and practical maximum data rates between 500kb/s and 2Mb/s, depending on conditions at a given site [20].

In Africa, a unique opportunity exists to install expansive wireless infrastructures due to a relative absence of equivalent wired infrastructures. Countries that skipped wired infrastructures due to inhibitive costs and unsympathetic geography, can use WiMAX to enhance their communication infrastructure in an inexpensive, decentralised, deployment-friendly and effective manner [20].

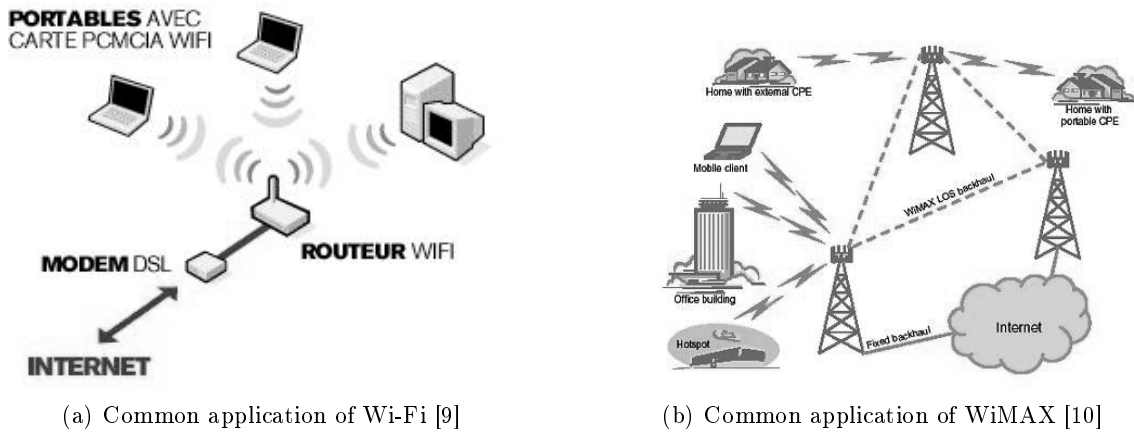


Figure 2.1: Common wireless network applications

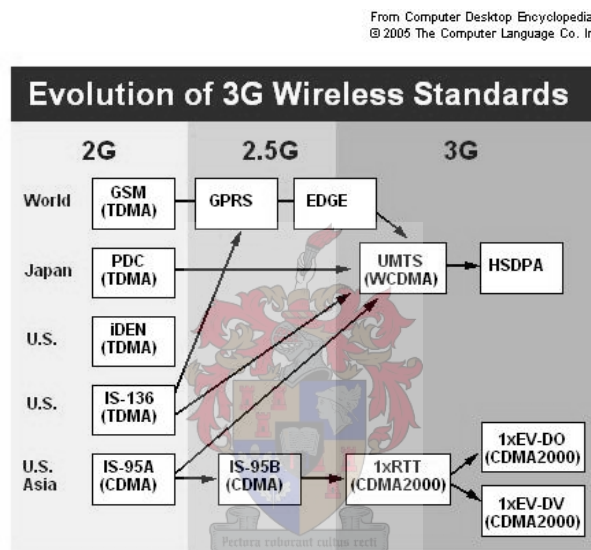


Figure 2.2: Evolution of 3G[15]

2.2.4 Cellular technologies

All of the following cellular technologies are services provided by third party service providers, namely Cell C, MTN and Vodacom. This needs to be taking into consideration when a decision on the final solution is made.

The different cellular technologies have evolved through various generations to the current 3G technology being implemented today. This evolution can be seen in Figure 2.2. Currently, cellular service providers in South Africa offer GSM and GPRS through to HSDPA services.

GPRS

General packet radio service (GPRS) is a mobile data service available to users of GSM mobile phones. It is often described as "2.5G", that is, a technology between the second (2G) and third (3G) generation of mobile telephony. It provides moderate speed data transfer by using TDMA channels in the GSM network.

The theoretical limit for packet switched data is approximately 160.0kb/s, using 8 time slots and CS-4. A realistic bit rate is 30-80kb/s, because it is only possible to use a maximum of 4

time slots for the downlink [7]. GPRS also has to contend with voice data for time slots and because voice data is prioritised over GPRS data, the effective speed for GPRS is considerably lower.

The maximum speed of a GPRS connection, as offered in 2003, is the same as a modem connection in an analogue wire telephone network, about 4-5kB/s, depending on the phone used. Latency for GPRS is high; a round-trip ping being typically about 600-700ms and often reaching a round-trip-time (RTT) of one second. GPRS has a lower priority than that of speech in cellular networks, and thus the quality of the connection varies greatly [7].

The above paragraph already eliminates GPRS as a potential candidate for the communication network, due to its high latency and poor link reliability. However, it is still included in the study for completeness. In South Africa, Vodacom offers GPRS access with speeds of up to 80kb/s [18].

EDGE

Enhanced data rates for GSM evolution, or EDGE, is a digital mobile phone technology which acts as a bolt-on enhancement to 2G and 2.5G GPRS networks. EDGE works in GSM networks, because EDGE is a superset to GPRS and can function on any network with GPRS deployed on it, provided the carrier implements the necessary upgrades. EDGE can carry data speeds of up to 236.8kb/s for 4 time slots, with a theoretical maximum of 473.6kb/s for 8 time slots, in packet mode [6].

The increased data rates of EDGE and its compatibility with existing GSM networks removes the concerns for coverage provided by cellular service providers. In South Africa, Vodacom offers EDGE access with speeds of up to 170kb/s [18].

3G

3G is short for third generation technology. The service associated with 3G provide the ability to transfer both voice data (a telephone call), and non-voice data (downloading information, exchanging email, and instant messaging). 3G uses a CDMA radio interface [2] and provides broad bandwidth and high speeds, with 384kb/s for mobile and 2Mb/s for stationary systems [5].

Due to the fact that 3G is not compatible with GSM networks, a new 3G infrastructure needs to be developed. Because this is still a relatively new technology in South Africa the coverage of current 3G networks are not as widespread as GSM networks. This causes 3G enabled devices to default to EDGE or GPRS mode when 3G coverage cannot be obtained. This could have a detrimental effect in the planned communication network, where certain TICs might not have 3G coverage. In South Africa, Vodacom offers 3G access with speeds of up to 384kb/s [18].

HSDPA

High-speed downlink packet access (HSDPA) is a new mobile telephony protocol and is sometimes referred to as a 3.5G technology. In this respect, HSDPA extends WCDMA in the same way that EV-DO extends CDMA2000. HSDPA provides a smooth evolutionary path for universal mobile telecommunications systems (UMTS) networks allowing for higher data capacity, with up to 14.4Mb/s in the downlink. It is an evolution of the WCDMA standard, designed to increase

the available data rate by a factor of 5 or more. HSDPA defines a new WCDMA channel, the high-speed downlink shared channel (HS-DSCH), that operates in a different way from existing WCDMA channels, but is only used for downlink communication.

A HSDPA upgrade to a network, is not a single step change, as there are different evolutions of this network. QPSK is the initial modulation scheme, however, in good radio conditions the introduction on 16QAM modulation will improve data throughput rates by approximately double that of QPSK. QPSK with 5 code allocation will typically offer up to 1.8Mb/s peak data rates. 16QAM with 5 codes will increase this to 3.6Mb/s. Additional codes, for example 10 or 15, can also be used to improve these data rates or extend the network capacity throughput significantly. Theoretically, HSDPA can give throughput up to 10.8Mb/s [11].

In South Africa, Vodacom offers HSDPA access with speeds of up to 1.8Mb/s [18].

2.2.5 Hardware bandwidths

Table 2.2 provides a list of connection bandwidths of some computer devices employing methods of data transport. These are listed by kb/s, Mb/s, or Gb/s as appropriate. They are also listed in bytes per second and are listed in order from lowest to highest bandwidth. Many of these figures are theoretical maximums, and various real-world considerations will result in a lower actual effective throughput [12].¹

Table 2.2: *Device bandwidths [12]*

Connection	Bits	Bytes
Wide area network		
DS1/T1	1.544 Mb/s	192.5 kB/s
OC-3/STM-1	155.52 Mb/s	19.44 MB/s
OC-12/STM-4	622.08 Mb/s	77.76 MB/s
OC-48/STM-16	2.448320 Gb/s	306.104 MB/s
OC-192/STM-64	9.953280 Gb/s	1.24416 GB/s
OC-768/STM-256	39.813120 Gb/s	4.97664 GB/s
BPL	2.7 Mb/s	337.5 kB/s
Wireless		
802.11 legacy 0.125	2000 kb/s	250 kB/s
802.11b DSSS 0.125	11 Mb/s	1.375 MB/s
802.11b+ non-standard DSSS 0.125	44.0 Mb/s	5.5 MB/s
802.11a 0.75	54.00 Mb/s	6.75 MB/s
802.11g DSSS 0.125	54.00 Mb/s	6.75 MB/s
802.11n	600 Mb/s	75 MB/s
802.16 (WiMAX)	70 Mb/s	8.75 MB/s
Mobile telephone interfaces		
GPRS upstream	28.8 kb/s	3.6 kB/s
GPRS downstream	57.6 kb/s	7.2 kB/s
EDGE	473.6 kb/s	59.2 kB/s
3G (stationary)	384 kb/s	48 kB/s
3G (mobile)	2 Mb/s	250 kB/s
HSDPA	10.8 Mb/s	1.35 MB/s

¹In telecommunications, 1kb/s = 1000 bit/s, not 1024 bit/s. Thus, all values in Table 2.2 use metric prefixes.

2.2.6 Comparing technologies

The different technologies are compared on a few basic aspects, such as maximum usable distance, security and speed. For a summary of the comparison see Table 2.3. This provides an overview of the different advantages and disadvantages of the various technologies².

Table 2.3: *Technology comparison*

Category	OC	BPL	Wi-Fi	WiMAX	GPRS	EDGE	3G	HSDPA
Distance	2km/hop		500m/hop	8km/hop	N/A	N/A	N/A	N/A
Wired	Yes	Yes	No	No	No	No	No	No
Digging	Yes	No	No	No	No	No	No	No
Security	Very good	Very good	Poor	Good	Very good	Very good	Very good	Very good
Expandable	Difficult	Easy	Easy	Easy	Highly *	Highly *	Highly *	Highly *
Cost	High	Low	Low	Low	Medium	Medium	High	High
SP	No	Yes	No	No	Yes	Yes	Yes	Yes
Max. speed	40Gb/s	2.7Mb/s	54Mb/s	70Mb/s	80kb/s	236.8kb/s	2Mb/s	10.8Mb/s
Real speed	155Mb/s	256kb/s	1Mb/s	2Mb/s	28.8kb/s U	170kb/s	384kb/s	1.8Mb/s
Real speed					57.6kb/s D			
LOS	No	No	Y/N	Y/N	No	No	No	No
Future proof	Good	Good	Good	Very good	Poor	Poor	Good	Good

2.2.7 Estimated technology latencies

The different technologies investigated in this study have varying capacities and operate at different speeds. This prompts us to look at the effective latencies of these technologies, in order to verify their applicability, and ability to satisfy the communication requirements of our network.

By substituting the device bandwidths into the model parameters established in Section 5.6, the resulting latencies can be calculated. The results are available in Tables 2.4, 2.5 and 2.6. The tables show the technology, the model used, associated data rates and the results from the latency models. These results include an indication of the stability, latency and the RTT.

It should be noted that the assumption of a dedicated channel (See Section 5.3) is not realistic in the case of cellular networks, where the channel is shared with thousands of other cellphone users. The cellular results are therefore not realistic, especially in the case of GPRS, which has to contend with voice data for time slots in the GSM frames. Due to the complex statistical models required to estimate network congestion and collisions with other users in a cellular network it will not be investigated any further.

The latency models also do not fully approximate the effects of the MAC protocols used, as it only includes their header sizes and does not take all wait times and control traffic overhead into account. This could result in optimistic latency results and real latencies could be higher.

²Expansion depends on the cellular coverage of the service provider (SP). Values from Table 2.3 marked with a (*) are subject to this constraint.

Table 2.4: Resulting latencies for wired technologies ($jitter_max = 0.125s$)

Technology	Model	Max.:Real Bandwidth	Stable	Max.:Real Latency	Max.:Real RTT
OC	1:1 Up	40Gbit/s : 155Mbit/s	Y:Y	62.5000ms : 62.5123ms	125.0001ms : 125.0245ms
	1:1 Down	40Gbit/s : 155Mbit/s	Y:Y	62.5000ms : 62.5123ms	125.0001ms : 125.0245ms
	N:1 Up	40Gbit/s : 155Mbit/s	Y:Y	62.5000ms : 62.5123ms	125.0001ms : 125.0245ms
	1:N Down	40Gbit/s : 155Mbit/s	Y:Y	62.5000ms : 62.5123ms	125.0001ms : 125.0245ms
	1:1:1 Up	40Gbit/s : 155Mbit/s	Y:Y	125.0001ms : 125.0330ms	250.0003ms : 250.0661ms
	1:1:1 Down	40Gbit/s : 155Mbit/s	Y:Y	125.0001ms : 125.0330ms	250.0003ms : 250.0661ms
	N:N:1 Up	40Gbit/s : 155Mbit/s	Y:Y	125.0002ms : 125.0517ms	250.0004ms : 250.1034ms
	1:N:N Down	40Gbit/s : 155Mbit/s	Y:Y	125.0001ms : 125.0330ms	250.0003ms : 250.0661ms
	1:1 Up (r)	40Gbit/s : 155Mbit/s	Y:Y	62.5001ms : 62.5135ms	125.0001ms : 125.0270ms
	1:1 Down (r)	40Gbit/s : 155Mbit/s	Y:Y	62.5001ms : 62.5135ms	125.0001ms : 125.0270ms
	BPL	1:1 Up	2.7MBit/s : 256kBit/s	Y:Y	63.2058ms : 70.1098ms
1:1 Down		2.7MBit/s : 256kBit/s	Y:Y	63.2058ms : 70.1098ms	126.4116ms : 140.2197ms
N:1 Up		2.7MBit/s : 256kBit/s	Y:Y	63.2074ms : 70.3018ms	126.4148ms : 140.6036ms
1:N Down		2.7MBit/s : 256kBit/s	Y:Y	63.2058ms : 70.1098ms	126.4116ms : 140.2197ms
1:1:1 Up		2.7MBit/s : 256kBit/s	Y:Y	126.9025ms : 145.7155ms	253.8050ms : 291.4310ms
1:1:1 Down		2.7MBit/s : 256kBit/s	Y:Y	126.9025ms : 145.7155ms	253.8050ms : 291.4310ms
N:N:1 Up		2.7MBit/s : 256kBit/s	Y:Y	128.0383ms : 167.3205ms	256.0767ms : 334.6411ms
1:N:N Down		2.7MBit/s : 256kBit/s	Y:Y	126.9025ms : 145.7155ms	253.8050ms : 291.4310ms
1:1 Up (r)		2.7MBit/s : 256kBit/s	Y:Y	63.2766ms : 70.8915ms	126.5531ms : 141.7829ms
1:1 Down (r)		2.7MBit/s : 256kBit/s	Y:Y	63.2766ms : 70.8915ms	126.5531ms : 141.7829ms

Table 2.5: Resulting latencies for wireless technologies ($jitter_max = 0.125s$)

Technology	Model	Max.:Real Bandwidth	Stable	Max.:Real Latency	Max.:Real RTT
Wi-Fi (11Mbps)	1:1 Up	11MBit/s : 1MBit/s	Y:Y	62.6911ms : 64.6157ms	125.3823ms : 129.2314ms
	1:1 Down	11MBit/s : 1MBit/s	Y:Y	62.6911ms : 64.6157ms	125.3823ms : 129.2314ms
	N:1 Up	11MBit/s : 1MBit/s	Y:Y	62.6913ms : 64.6303ms	125.3825ms : 129.2606ms
	1:N Down	11MBit/s : 1MBit/s	Y:Y	62.6911ms : 64.6157ms	125.3823ms : 129.2314ms
	1:1:1 Up	11MBit/s : 1MBit/s	Y:Y	125.5022ms : 130.5726ms	251.0045ms : 261.1452ms
	1:1:1 Down	11MBit/s : 1MBit/s	Y:Y	125.5022ms : 130.5726ms	251.0045ms : 261.1452ms
	N:N:1 Up	11MBit/s : 1MBit/s	Y:Y	125.7691ms : 134.0002ms	251.5383ms : 268.0004ms
	1:N:N Down	11MBit/s : 1MBit/s	Y:Y	125.5022ms : 130.5726ms	251.0045ms : 261.1452ms
	1:1 Up (r)	11MBit/s : 1MBit/s	Y:Y	62.7103ms : 64.8289ms	125.4206ms : 129.6577ms
	1:1 Down (r)	11MBit/s : 1MBit/s	Y:Y	62.7103ms : 64.8289ms	125.4206ms : 129.6577ms
Wi-Fi (54Mbps)	1:1 Up	54MBit/s : 2MBit/s	Y:Y	62.5389ms : 63.5542ms	125.0778ms : 127.1085ms
	1:1 Down	54MBit/s : 2MBit/s	Y:Y	62.5389ms : 63.5542ms	125.0778ms : 127.1085ms
	N:1 Up	54MBit/s : 2MBit/s	Y:Y	62.5389ms : 63.5579ms	125.0778ms : 127.1157ms
	1:N Down	54MBit/s : 2MBit/s	Y:Y	62.5389ms : 63.5542ms	125.0778ms : 127.1085ms
	1:1:1 Up	54MBit/s : 2MBit/s	Y:Y	125.1022ms : 127.7731ms	250.2045ms : 255.5462ms
	1:1:1 Down	54MBit/s : 2MBit/s	Y:Y	125.1022ms : 127.7731ms	250.2045ms : 255.5462ms
	N:N:1 Up	54MBit/s : 2MBit/s	Y:Y	125.1559ms : 129.3467ms	250.3119ms : 258.6934ms
	1:N:N Down	54MBit/s : 2MBit/s	Y:Y	125.1022ms : 127.7731ms	250.2045ms : 255.5462ms
	1:1 Up (r)	54MBit/s : 2MBit/s	Y:Y	62.5428ms : 63.6601ms	125.0856ms : 127.3201ms
	1:1 Down (r)	54MBit/s : 2MBit/s	Y:Y	62.5428ms : 63.6601ms	125.0856ms : 127.3201ms
WiMAX	1:1 Up	70MBit/s : 2MBit/s	Y:Y	62.5269ms : 63.4430ms	125.0537ms : 126.8861ms
	1:1 Down	70MBit/s : 2MBit/s	Y:Y	62.5269ms : 63.4430ms	125.0537ms : 126.8861ms
	N:1 Up	70MBit/s : 2MBit/s	Y:Y	62.5269ms : 63.4459ms	125.0537ms : 126.8918ms
	1:N Down	70MBit/s : 2MBit/s	Y:Y	62.5269ms : 63.4430ms	125.0537ms : 126.8861ms
	1:1:1 Up	70MBit/s : 2MBit/s	Y:Y	125.0726ms : 127.5502ms	250.1451ms : 255.1003ms
	1:1:1 Down	70MBit/s : 2MBit/s	Y:Y	125.0726ms : 127.5502ms	250.1451ms : 255.1003ms
	N:N:1 Up	70MBit/s : 2MBit/s	Y:Y	125.1139ms : 129.1148ms	250.2279ms : 258.2297ms
	1:N:N Down	70MBit/s : 2MBit/s	Y:Y	125.0726ms : 127.5502ms	250.1451ms : 255.1003ms
	1:1 Up (r)	70MBit/s : 2MBit/s	Y:Y	62.5296ms : 63.5377ms	125.0591ms : 127.0753ms
	1:1 Down (r)	70MBit/s : 2MBit/s	Y:Y	62.5296ms : 63.5377ms	125.0591ms : 127.0753ms

Table 2.6: Resulting latencies cellular technologies ($jitter_max = 0.125s$)

Technology	Model	Bandwidth Max.:Real	Stable	Latency Max.:Real	RTT Max.:Real
GPRS	1:1 Up	80KBit/s : 28.8KBit/s	Y:Y	94.7036ms : 172.2752ms	189.4073ms : 344.5504ms
	1:1 Down	80KBit/s : 57.6KBit/s	Y:Y	94.7036ms : 109.1149ms	189.4073ms : 218.2297ms
	N:1 Up	80KBit/s : 28.8KBit/s	Y:Y	98.4466ms : 232.6783ms	196.8932ms : 465.3566ms
	1:N Down	80KBit/s : 57.6KBit/s	Y:Y	94.7036ms : 109.1149ms	189.4073ms : 218.2297ms
	1:1:1 Up	80KBit/s : 28.8KBit/s	Y:Y	210.7355ms : 449.5686ms	421.4710ms : 0.8991s
	1:1:1 Down	80KBit/s : 57.6KBit/s	Y:Y	210.7355ms : 251.3304ms	421.4710ms : 0.5027s
	N:N:1 Up	80KBit/s : 28.8KBit/s	N:N	-1.3857s : 52.9527ms	-2.7714s : 105.9054ms
	1:N:N Down	80KBit/s : 57.6KBit/s	Y:Y	210.7355ms : 251.3304ms	421.4710ms : 0.5027s
	1:1 Up (r)	80KBit/s : 28.8KBit/s	Y:Y	98.2967ms : 187.6965ms	196.5935ms : 375.3929ms
	1:1 Down (r)	80KBit/s : 57.6KBit/s	Y:Y	98.2967ms : 114.5610ms	196.5935ms : 229.1221ms
EDGE	1:1 Up	236.8KBit/s : 170KBit/s	Y:Y	72.6779ms : 76.8629ms	145.3557ms : 153.7258ms
	1:1 Down	236.8KBit/s : 170KBit/s	Y:Y	72.6779ms : 76.8629ms	145.3557ms : 153.7258ms
	N:1 Up	236.8KBit/s : 170KBit/s	Y:Y	73.0242ms : 77.5624ms	146.0484ms : 155.1248ms
	1:N Down	236.8KBit/s : 170KBit/s	Y:Y	72.6779ms : 76.8629ms	145.3557ms : 153.7258ms
	1:1:1 Up	236.8KBit/s : 170KBit/s	Y:Y	151.4033ms : 162.4397ms	302.8067ms : 324.8795ms
	1:1:1 Down	236.8KBit/s : 170KBit/s	Y:Y	151.4033ms : 162.4397ms	302.8067ms : 324.8795ms
	N:N:1 Up	236.8KBit/s : 170KBit/s	Y:Y	178.5031ms : 216.6599ms	357.0062ms : 433.3199ms
	1:N:N Down	236.8KBit/s : 170KBit/s	Y:Y	151.4033ms : 162.4397ms	302.8067ms : 324.8795ms
	1:1 Up (r)	236.8KBit/s : 170KBit/s	Y:Y	73.7326ms : 78.3729ms	147.4652ms : 156.7458ms
	1:1 Down (r)	236.8KBit/s : 170KBit/s	Y:Y	73.7326ms : 78.3729ms	147.4652ms : 156.7458ms
3G	1:1 Up	2MBit/s : 384KBit/s	Y:Y	63.6711ms : 68.6982ms	127.3422ms : 137.3963ms
	1:1 Down	2MBit/s : 384KBit/s	Y:Y	63.6711ms : 68.6982ms	127.3422ms : 137.3963ms
	N:1 Up	2MBit/s : 384KBit/s	Y:Y	63.6755ms : 68.8249ms	127.3511ms : 137.6498ms
	1:N Down	2MBit/s : 384KBit/s	Y:Y	63.6711ms : 68.6982ms	127.3422ms : 137.3963ms
	1:1:1 Up	2MBit/s : 384KBit/s	Y:Y	128.0073ms : 141.0066ms	256.0145ms : 282.0132ms
	1:1:1 Down	2MBit/s : 384KBit/s	Y:Y	128.0073ms : 141.0066ms	256.0145ms : 282.0132ms
	N:N:1 Up	2MBit/s : 384KBit/s	Y:Y	129.5907ms : 153.1732ms	259.1813ms : 306.3463ms
	1:N:N Down	2MBit/s : 384KBit/s	Y:Y	128.0073ms : 141.0066ms	256.0145ms : 282.0132ms
	1:1 Up (r)	2MBit/s : 384KBit/s	Y:Y	63.7887ms : 69.3317ms	127.5774ms : 138.6633ms
	1:1 Down (r)	2MBit/s : 384KBit/s	Y:Y	63.7887ms : 69.3317ms	127.5774ms : 138.6633ms
HSDPA	1:1 Up	10.8MBit/s : 1.8MBit/s	Y:Y	62.7162ms : 63.8018ms	125.4324ms : 127.6035ms
	1:1 Down	10.8MBit/s : 1.8MBit/s	Y:Y	62.7162ms : 63.8018ms	125.4324ms : 127.6035ms
	N:1 Up	10.8MBit/s : 1.8MBit/s	Y:Y	62.7164ms : 63.8073ms	125.4327ms : 127.6145ms
	1:N Down	10.8MBit/s : 1.8MBit/s	Y:Y	62.7162ms : 63.8018ms	125.4324ms : 127.6035ms
	1:1:1 Up	10.8MBit/s : 1.8MBit/s	Y:Y	125.5546ms : 128.3433ms	251.1092ms : 256.6866ms
	1:1:1 Down	10.8MBit/s : 1.8MBit/s	Y:Y	125.5546ms : 128.3433ms	251.1092ms : 256.6866ms
	N:N:1 Up	10.8MBit/s : 1.8MBit/s	Y:Y	125.8268ms : 130.1206ms	251.6536ms : 260.2411ms
	1:N:N Down	10.8MBit/s : 1.8MBit/s	Y:Y	125.5546ms : 128.3433ms	251.1092ms : 256.6866ms
	1:1 Up (r)	10.8MBit/s : 1.8MBit/s	Y:Y	62.7378ms : 63.9325ms	125.4757ms : 127.8651ms
	1:1 Down (r)	10.8MBit/s : 1.8MBit/s	Y:Y	62.7378ms : 63.9325ms	125.4757ms : 127.8651ms

2.2.8 Summary

In this section, different communication technologies were investigated. Their merits and shortcomings were discussed, with a focus on an ITS implementation and low latency.

While wired technologies tend to have high bandwidth, they are limited by physical factors, such as digging required to bury the wires underground. This could increase costs involved with deploying wide area communication infrastructures with wired technologies.

Cellular technologies provide moderate bandwidth, whilst covering wide areas. This would make communication infrastructures deployed over cellular technologies easily expandable. These technologies are presently only provided by third party service providers. This could increase the costs involved with deploying an ITS system over such a network, as large amounts of data could be transmitted over time.

Wireless technologies provide moderate to high bandwidth, and are easily expandable. With high gain, directional antennas, these network could easily cover wide areas. These networks can also be deployed independently, without a third party service provider. This makes wireless networks an attractive solution for deploying an ITS communication network.

Tables 2.4, 2.5 and 2.6 list theoretical latencies for the different communication technologies, which was obtained using latency models developed in Chapter 5. The results placed a minimum bandwidth requirement, of higher than 85kb/s, on communication technologies that would be able to facilitate an ITS system, as envisioned in the telecommunications requirements mentioned in Section 2.1.3. This discounted GPRS as a potential technology candidate for the communication system.

In the following section, a communication network is conceptualised that would be able to provide the communication backbone required by the CoCT.

2.3 Proposed ITS communication network

2.3.1 Overview

In the previous section various communication technologies were discussed, and their limitations were investigated. In this section, additional functionality that could be added to the communication network for the ITS system is discussed. A communication network is also conceptualised, describing a potential network architecture. Finally, a conclusion is drawn up on the proposed solution for the CoCT.

2.3.2 Additional network functionality

If the employed network has additional bandwidth, then additional functionality and services can be accommodated. This section will briefly look at what services could be added to the network to enhance its value, and what types of constraints they will place on the network.

Some of these services are time critical, and require quick reaction times, while other services can afford long delays in the network. Some of these services cannot afford any data being lost, while other services can accommodate data losses.

Time servers

Time servers are critical to synchronise the system clocks of the TICs, and could consume a large amount of the available bandwidth. Time servers are generally difficult to employ on large wireless networks due to varying latencies in the network.

These systems are time critical and losses should be kept to a minimum.

Closed circuit television

Closed circuit television (CCTV) cameras capture images at certain frame rates and these are transmitted back to the traffic directorate offices of the CoCT, where intersections can be monitored via video. Depending on the frame rate and the quality of the images, these systems use varying amounts of bandwidth. In general, the packets generated by low frame-rate CCTV are reasonably small (in the kB range). These should experience low latencies, and will allow CCTV systems to be employed on this network.

These systems are time critical, but losses can be tolerated to an extent.

Voice and video over IP

Employees of the CoCT can employ voice over IP (VoIP) telecommunication services to communicate at no cost, within coverage of the communication network. This could drastically reduce the expenditure of the municipality. Skype has recently announced that it could be developing VoIP cellular phones, and this could allow municipal employees away from the office to communicate with employees at the office, at no cost.

If the CCTV systems are replaced with video IP (VIP) systems, then these systems could also operate over the same network.

These systems are time critical, but losses can be tolerated to an extent.

Variable messaging signs

By adding functionality to the network, to carry messages for variable message signs (VMS) in and around the CoCT, the public transport system can be greatly improved. VMS can be used to provide information regarding traffic intensity on various routes, or to inform the public of accidents and advise alternative routes. They can also be employed at bus stops to inform customers when the next bus will arrive, or can even be employed in busses, to let customers know where a bus is heading.

This service has a low priority, and is not time critical. It can also tolerate losses because information updates to VMS signs would have long periods in between updates and information would be long lived. It can, therefore, be given a low priority in the network, minimising its effect on other networked services. It is therefore a viable addition to the network.

Internet access

Internet access could be given to technicians working on hardware on the road, or could even be supplied to customers at a fee. The amount of bandwidth available for Internet usage should be limited, minimising the effect of Internet traffic on other systems running on the network.

If Internet access is provided, security measures should be taken to secure the network from unwanted access by malicious users. Initially, the communication network will not cover the entire metropolitan area of the CoCT, so access would be limited to certain areas. In the future, the network could be expanded to cover the entire area of the CoCT and service all of its citizens.

It should also be noted that the bandwidth available for Internet access would be shared by multiple users, and speeds could drop drastically if too many users try to use the available bandwidth. Careful monitoring and control should be practised to avoid this eventuality.

2.3.3 Conceptual network design

Overview

From the results obtained during this study, a solution has been drafted that can be effectively used to satisfy the basic needs of the CoCT, with the possibility of adding additional functionality. It is our belief that the solution proposed in this section is the most versatile solution to provide the most benefit to the CoCT.

During discussions with the CoCT, plans to deploy a large, fibre optic backbone throughout the city were discussed. Therefore, the solution proposed in this section assumes that the CoCT already has such a backbone available.

The proposed solution comprises of a fibre optic backbone connected to the instation, or central controller. At strategic locations along the backbone, wireless extensions would be made via Wi-Fi nodes. These wireless nodes can then branch out to all the surrounding outstations or TICs. For long distances, WiMAX nodes can be used to extend the reach of the wireless network. This would achieve a wireless mesh or ad hoc wireless network throughout the city.

The goal for this particular approach is to maximise available bandwidth and minimise wireless hops. With this approach, all reachable nodes will have a minimum number of wireless hops to the high bandwidth backbone, that in turn is directly connected to the instation and possibly the Internet.

From the Old to the New

The old communication network used a traditional circuit switched network, and used hybrid switches to connect the appropriate circuits together to provide communication to a desired destination.

The proposed communication network will employ a packet switched network, where the TIC themselves become part of the network, and can route data to other TICs. The difference between the old and new network topologies are displayed in Figure 2.3.

The new network topology not only removes some redundant hardware, for instance the switches in the old topology, but it also allows the network to expand with little or no effort, as a new TIC will simply attach itself to the network.

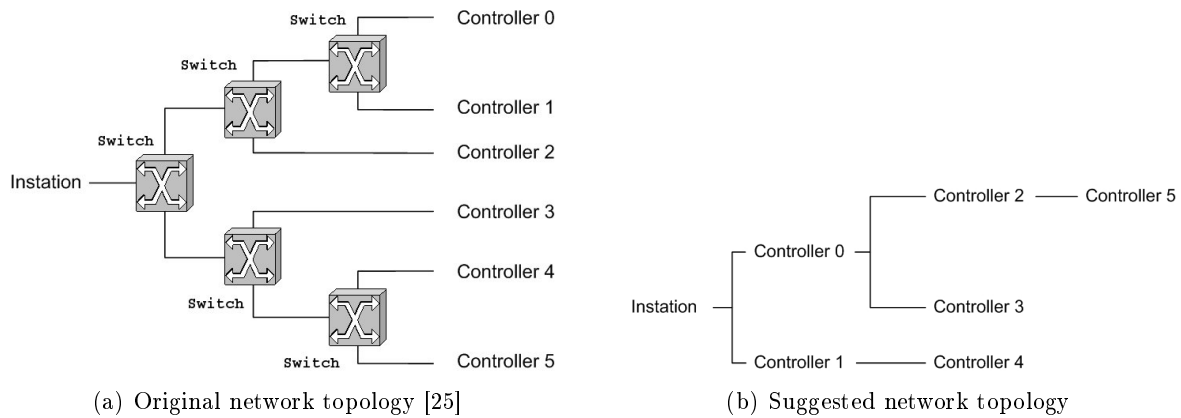


Figure 2.3: Comparing the old and suggested network topologies

Communication protocols

Communication will be achieved through two protocol groups, namely MAC and routing protocols. The MAC protocols would be the 802.11, 802.16, SONET and Ethernet protocols. Running on top of these protocols would be the TH(O)RP routing protocol developed as part of this investigation.

The TH(O)RP routing protocol follows a hierarchical tree approach, with the instation at the root of the tree and the outstations as nodes and leaves of the tree. This protocol was developed to minimise control traffic overhead, whilst providing fast, reliable communication to all nodes in the network. It was also developed with a large network in mind, up to hundreds of nodes, to make the resulting network easily expandable.

Network architecture

The network will consist of an optical carrier, high bandwidth, backbone. The wireless extensions would be implemented via Wi-Fi, for short distance and enclosed areas, and WiMAX, for long distance back haul needs. The wireless nodes could have a multi-radio configuration, to avoid collisions in the communication medium and improve network throughput. The WiMAX and Wi-Fi backbone nodes would use high gain, directional antennas, to maximise the link qualities of the backbone, whilst the remaining nodes would use omni-directional antennas to allow maximum coverage of their surrounding areas. This would create hot spots through the CoCT, allowing access to municipal and government employees and ultimately Internet access to the residents of the CoCT. An abstract view of the proposed network can be seen in Figure 2.4.

Multi-radio configuration and directional antennas

As mentioned before, a multi-radio approach could be followed. This allows for better throughput due to less collisions in the shared communication medium. The two radios would be able to transmit simultaneously due to the fact that they use different frequencies, namely the 2.4GHz and 5.8GHz bands if the 802.11b/g and 802.11a radios are employed. An illustration of this configuration can be seen in Figure 2.5. Directional antennas can also be used to increase the network throughput.

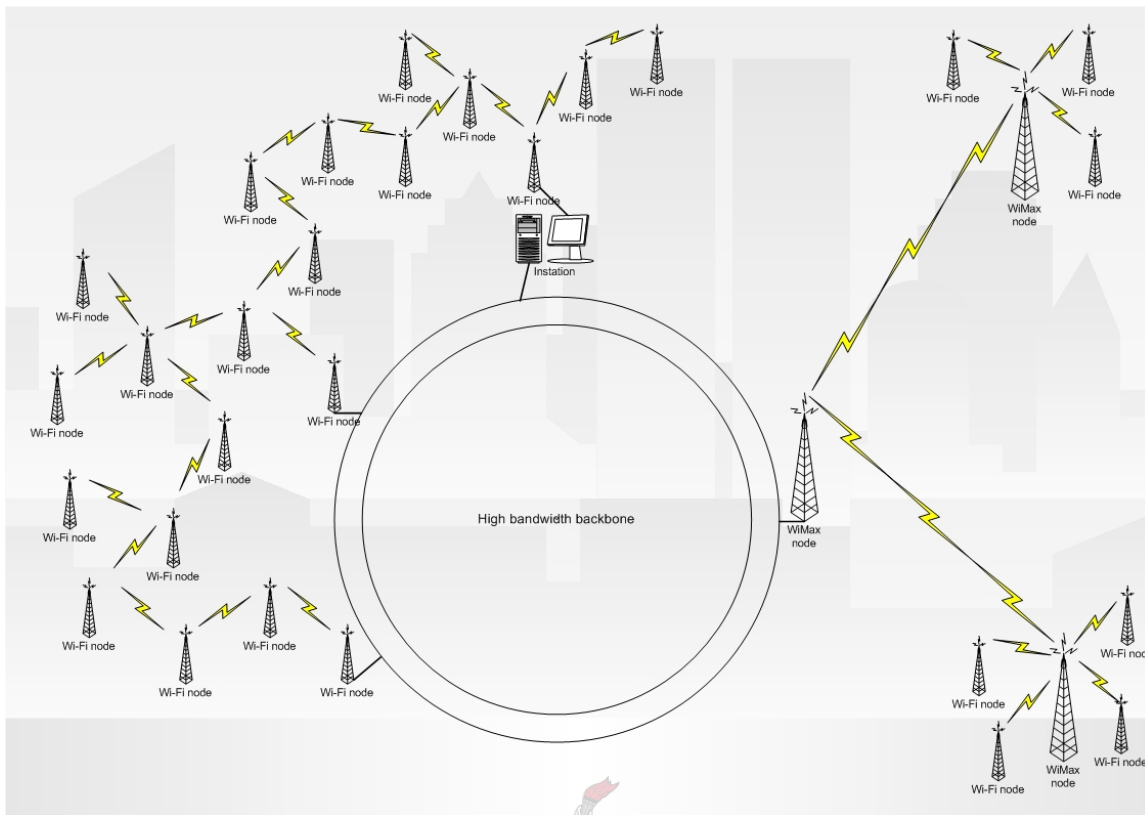


Figure 2.4: *Abstraction of network architecture*

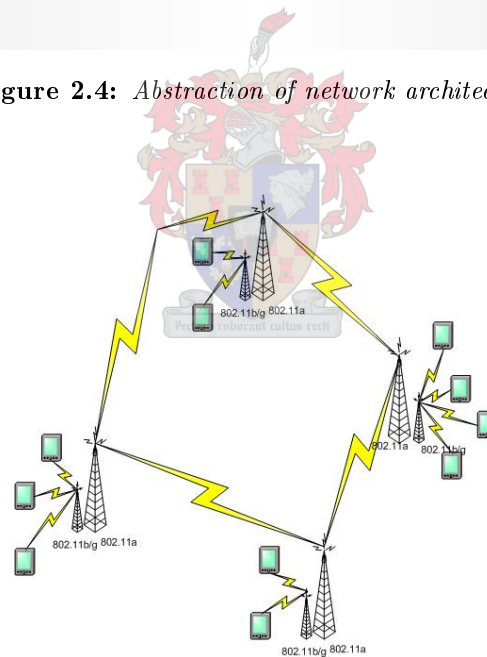


Figure 2.5: *Abstraction of a multi-radio configuration*

2.3.4 Conclusion

By moving away from any third party service provider, the CoCT would be able to become its own virtual private network (VPN), and will have a high bandwidth, metropolitan LAN at its disposal. This could be extended to enable fast, inexpensive communication between different departments of the city management, and can be expanded to deliver further services such as fast response to disasters and better cooperation and coordination between different emergency departments. If an effective ITS system is employed, it could be used to reduce traffic congestion and to promote a safe, reliable public transport system in the CoCT.

The communication network could also allow instant connectivity to the Internet anywhere in the covered area. If there is enough bandwidth in the new communication network, services such as VoIP and VIP can also be included, making communicating between departments in the CoCT virtually free.

The possible benefits of a new, up to date, communication network are astounding, and the consequences far reaching. Especially with the 2010 Soccer World Cup Tournament coming to South Africa, an ITS system that is on par with global standards would be essential.

2.4 Summary

This chapter presented a feasibility study on various communication technologies that could be used to implement a communication network to facilitate an ITS system. A communication network is also conceptualised, describing a possible network architecture.

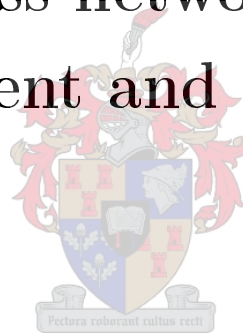
A BPL network would almost certainly be a potential candidate to provide a communication network able to deploy large scale ITS systems, due to the existing infrastructure provided by power distribution lines. This technology was, however, included in the study too late to be considered.

In the next part of this document, a multi-hop routing protocol is developed that could be used to manage the communication network suggested in this study.



Part II

Ad hoc wireless networking protocol development and evaluation



Chapter 3

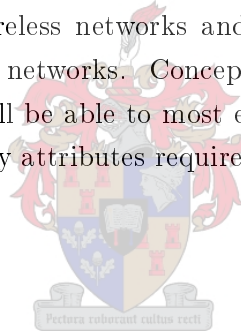
Theoretical background

3.1 Introduction

A communications network is desired to control and monitor the status of the traffic intersection controllers (TIC) of the City of Cape Town (CoCT). This same network would also be used to implement an ITS in the CoCT. A proposal was made in the previous chapter that the communications network could be implemented via a wireless mesh or ad hoc wireless network.

This chapter looks at ad hoc wireless networks and the different categories of multi-hop routing protocols that manage these networks. Concepts from these protocols are then used to develop a routing protocol that will be able to most effectively manage the communications network envisioned for the CoCT. Key attributes required of the protocol are:

- low latency
- self-configuring and self-healing
- scalability
- stability



Section 3.3 gives an overview of the different multi-hop routing protocols and evaluates them on the above criteria.

3.2 Ad hoc wireless networks: a routing problem

Ad hoc wireless networks are infrastructure-less networks, with routing and network management done in a distributed manner [39]. For example routing messages through these networks is done in a collaborative manner as individual nodes rely on the knowledge of other nodes to route message through the network. This requires nodes to be more intelligent than traditional network nodes and nodes can take on the role of a host for transmitting and receiving data as well as routing packets for other nodes [39].

Subsequently the routing protocols used in these networks are more complex than those used in traditional wired networks, as they have to route information over multiple hops in a network topology that can change regularly and without notice. These protocols need to keep track of routes, renewing routes that have become stale and unusable, optimising routes when shorter options become available and avoiding route loops. In some instances the maintenance

of routes in the network could consume all of the available bandwidth, or they can take long to be established, increasing the latency of a network. Various protocols have been developed to solve the routing problems associated with ad hoc wireless networks and some of these protocols are discussed in the following section.

3.3 Multi-hop routing protocols

Multi-hop routing protocols are classified as those protocols that manage ad hoc wireless networks. Various approaches have been followed in an attempt to effectively solve the routing problems associated with ad hoc networks. These protocols can be grouped into three distinct categories. The first category are the proactive protocols that attempt to maintain routes to all destinations in the network at all times. The second category are the reactive protocols that only set up routes when they are required. The final category are the hybrid protocols that combine concepts from proactive and reactive routing protocols.

The three categories can be further divided into two sub-categories, namely flat and hierarchical routing protocols. Each of these categories and sub-categories have their own advantages and disadvantages. Examples of these protocols are discussed below.

3.3.1 Proactive vs. reactive vs. hybrid approach

As stated above, there are three main categories for ad hoc routing protocols. The protocols in the proactive category attempt to maintain a clear understanding of the network topology at all times. They update their understanding of the network with periodic updates. Advantages of these protocols is that there is little or no latency when an application starts transmitting information because the routes of the network have already been established. These protocols also rarely have stale routes due to their periodic updates, so routes tend to be reliable. The area where proactive protocols fail, is with the large amounts of control traffic overhead generated through the periodic updates. This becomes a particularly big problem when networks grow and the periodic updates also increase in size, congesting the network. In quiet networks with infrequent communication, the proactive maintenance of routes wastes bandwidth as the routes are un-utilised.

Reactive routing protocols maintain a minimal amount of information about a network and only construct routes when a node requests such a route. This has the advantage of minimising the amount of control traffic required to maintain a network and also does not waste bandwidth in networks with infrequent communication. The disadvantage with reactive protocols is an initial delay that is experienced while routes are being constructed, which is experienced as latency by an application using a reactive protocol. This problem can be compounded when frequent communications are required, meaning routes have to be set up often.

A combination of proactive and reactive routing protocols are hybrid protocols. These protocols aim to combine the advantages of proactive and reactive protocols in order to achieve an optimal solution. Usually hybrid protocols use a proactive approach inside a segment or zone of a network and reactively set up routes to nodes outside that zone. The advantage of this approach is that nodes within a zone can be quickly reached, while nodes outside the zone can be reached without overwhelming the network with excessive control traffic. A disadvantage of this approach is that routes between nodes in separate zones are often sub-optimal.

The following section will look at various multi-hop routing protocols. While many more such routing protocols exist they are not discussed here.

3.3.2 Flat routing

Flat routing protocols view a network as a singular entity. This approach does not distinguish between network nodes and view all nodes as equal, which simplifies the management of these networks. Their flat routing approach do however make it difficult for these protocols to scale to large networks.

Destination-sequence distance vector

The destination-sequence distance vector [42, 39] routing protocol is an enhanced version of the distributed Bellman-Ford algorithm, with each node maintaining a routing table that contains the shortest distance and the first node on the shortest path to every other node in the network. It incorporates periodic updates with increasing sequence number tags to enable loop free routing and fast route convergence.

Because this is a table driven protocol routes to all destinations are available at all nodes at all times. The tables maintained by DSDV are periodically sent to neighbouring nodes in order to keep an up-to-date view of the network topology. DSDV also forwards its tables when it observes significant changes in the local topology. While DSDV has the advantage of always having routes available to all network nodes it does suffer from excessive control overhead proportional to the number of nodes. This severely limits the scalability of DSDV in ad hoc wireless networks.

DSDV is one of the first protocols proposed for ad hoc wireless networks and is an example of a proactive, table driven protocol.

Optimised link state routing

The optimised link state routing (OLSR) protocol [26, 39] is an optimisation of the classical link state algorithm, tailored to the requirements of mobile wireless LANs. OLSR routes information through a network using controlled flooding. A subset of nodes are appointed as multi-point relays (MPR). Only MPR nodes are allowed to forward broadcast messages during the flooding process, which brings a substantial optimisation over traditional flooding where every node in a network forwards a broadcasted message.

OLSR also only allows MPR nodes to generate link state information, reducing the amount of control traffic in a network. As a final optimisation OLSR distributes partial link state information through the network and this information can be used to calculate routes. OLSR calculates route costs in terms of hops and is well suited to dense, large networks due to the MPR concept.

OLSR is an example of a proactive, table driven protocol.

Topology dissemination based on reverse-path forwarding

The topology dissemination based on reverse-path forwarding (TBRPF) protocol [40] is another example of a proactive, link-state routing protocol that is designed for mobile ad hoc networks. TBRPF provides hop-by-hop routing along shortest paths to each destination in a network.

Each TBRPF node computes a source tree based on partial topology information, using a modification of Dijkstra's algorithm. Each node can report only parts of its source tree in order to reduce control traffic overhead. In highly mobile networks nodes can report additional topology information, improving the protocols robustness. Differential updates are used to reduce control traffic overhead, where only changes in the network topology are reported.

TBRPF is another example of a proactive, table driven protocol.

Ad hoc on-demand distance vector

The ad hoc on-demand distance vector (AODV) protocol [41, 39] is an example of a reactive routing protocol, where routes are only set up as they are required. AODV provides quick adaptation to dynamic link conditions and due to its reactive nature it offers low memory- and control traffic overhead. It can determine uni-cast routes to destinations within an ad hoc network and with the use of destination sequence numbers it avoids routing loops. It does, however, suffer from an initial latency while routes are being constructed.

3.3.3 Hierarchical routing

Hierarchical routing protocols segment a network into subnetworks, effectively decreasing the size of a network with respect to the protocol. This allows hierarchical protocols to scale to large networks, as the protocol only maintains a subset of the entire network. Hierarchical protocols can be complex, and automatically establishing and maintaining a hierarchy could prove difficult [21].

ZRP

The zone routing protocol (ZRP) [36, 39] is a hybrid between proactive and reactive routing protocols. It pro-actively maintains routes to nodes located inside a zone using an intra-zone routing protocol, and reactively sets up routes to nodes outside this zone using an inter-zone routing protocol [39]. By segmenting the network into zones ZRP can scale to larger networks as nodes are not required to maintain routes to all nodes in the network. By adjusting the zone radius parameter, ZRP can become more proactive (increase radius) or reactive (decrease radius). ZRP defines gateway nodes that connect adjacent zones, and the gateway nodes typically reside higher up the hierarchy with the nodes inside a zone lower in the hierarchy.

ZRP is an example of a hybrid routing protocol, with a hierarchical property.

OrderOne Networks

The OrderOne networks protocol [29] automatically derives a hierarchy from the network and organises this hierarchy in a tree like structure. It derives a hierarchy by using a modified version of the DSDV protocol and selecting the node with the most routes through it as its parent in the tree hierarchy. After the hierarchy has been established the protocol uses the high-speed propagation path (HSPP) to distribute routing information through the hierarchy.

This protocol is able to scale to potentially very large networks, as the tree hierarchy segments the network into smaller parts. By utilising the tree hierarchy this protocol is also able to facilitate route discovery with minimal flooding. Suboptimal routes and the hierarchy is pro-actively maintained and optimal routes are reactively set up. By having pro-actively maintained

routes, even though they are suboptimal, this protocol is able to remove the initial delay when setting up routes between nodes. Then through the reactive optimising process these routes converge to optimal routes.

3.4 Conclusion

Various approaches have been developed to manage ad hoc wireless networks, each having its own strengths and weaknesses. Most of the methods mentioned above, attempt to optimise routes to all destinations in a network. This is, however, excessive for the network developed in this project, as only optimal routes are required between a central entity and a large number of nodes, namely the instation and outstations of the CoCT. Due to the size of the envisioned network, attempting to establish optimal routes between all nodes would introduce excessive control overhead, consuming available bandwidth and increasing network latency. The size of the network is particularly a problem to the flat routing protocols.

The one category of protocols that showed promise was the hierarchical protocols, and more specifically the OdrOne network protocol [29]. Because of the nature of the communications that will occur in the proposed network, a hierarchical approach with the central controller at the top of the hierarchy will organise the network sensibly without complex routing algorithms. It was therefore decided to model the protocol developed in this thesis around similar concepts as those found in [29], but sufficiently adapted for the particular problem at hand.

The protocol that was developed to manage the proposed network for the CoCT is discussed in the next chapter.



Chapter 4

The tree hierarchy (open) routing protocol (TH(O)RP)

4.1 Introduction

The name TH(O)RP was derived from the tree hierarchy maintained by the routing protocol, and the (open) addition hails from the use of the open source tools and projects that made the protocol possible. The (O) in the acronym for TH(O)RP also acknowledges the OrderOne Networks Protocol[29] which automatically reveals a hierarchy from the network, and TH(O)RP uses similar concepts to construct its own tree hierarchy.

4.2 Design goals

The design goals of TH(O)RP are focused around addressing challenges identified during analysis of the communication network proposed in Section 2.3, as well as satisfying the telecommunication requirements mentioned in Section 2.1.3. The major challenges are:

- **Ad hoc deployment:** The communication system will be designed to be deployed quickly and on an **ad hoc** basis. These networks require automatic configuration.
- **Automatic operation:** The nodes of the communication system are required to operate **automatically**, without any human input. Because the network topology will be dynamic, with new nodes added and links being broken from time to time, the system is required to be self-configuring and self-healing.
- **Scale of proposed network:** The communication network will be large, thus the routing protocol will be required to be **scalable**. The network size is attributed to the large number of nodes planned at signalised traffic intersections in the CoCT.
- **Minimum latency requirement:** There is a minimum latency requirement placed on the network by the SCOOT system, which employs a TRAFX protocol using UTC messaging. The routing protocol will be required to have a **low latency**. The largest factors in the resulting latency is the 'length' of routes between nodes, average packet loss rate and other network traffic.

- **Interference:** There are two major sources of interference, namely electronic and physical interference. Electronic interference is due to other bandwidth users and electronic devices, such as microwaves and radios. Physical interference is due to the busy environment, namely buildings, vehicles and atmospheric interference, in which the communication system is implemented. This will disrupt the line of sight (LOS) between nodes in the network. The protocol will need to handle interference by keeping accurate approximations of the link status and qualities between nodes. This stresses the need for an accurate **link metric** that will take multiple factors into account when determining the quality of a link. Some of these factors will be the available bandwidth, average error rate and whether a link is uni- or bi-directional. With an accurate approximation of the link quality between nodes, the routing protocol can calculate optimum routes based on this metric, versus only using hop-count, and will be able to route around problem areas in the network.
- **Stability:** Because of the critical nature of the proposed application, the system is required to be **stable** and should be resilient to network losses.
- **Security:** Wireless networks are more prone to physical security threats than fixed cable networks [27]. Other security considerations are robustness against denial of service (DoS) attacks and man-in-the-middle attacks. These attacks can be addressed by using timestamps, public key distribution systems, secure socket layer (SSL) and secure transport layer (STL) techniques. If, in the future, the network is expanded to provide basic Internet access, it will need to be **secured**. This would make up the additional functionality of the protocol and could be implemented using plug-ins.
- **Additional functionality:** The system will be required to be highly **expandable**, providing additional services such as variable message signs (VMS). The system might also be connected to the Internet in the future, or use different interfaces to connect to different networks. To avoid the need to modify the core of the routing protocol, the protocol will be modular in design and contain a plug-in interface. The additional features can then be implemented via plug-ins.

In summary, the primary goals of the protocol were to enable automatic configuration and recovery, and to promote low latency, stability and scalability. It also aims to optimise the routes between the SCOOT instation and the outstations of the ITS communication network, namely the TICs.

4.3 Why a tree?

When ad hoc wireless networks become large the traditional flat routing algorithms cannot scale, so the need to segment the network by means of a hierarchy becomes essential. This abstraction minimises the size of a network with regards to the routing protocol, as it only manages a subset of the entire network, allowing it to scale to larger networks.

A tree is a naturally hierarchical entity, with a natural progression from higher to lower levels and vice versa. This property allows us to elegantly represent the hierarchy of a network and it allows us to define routes between any two points in the network. The use of a tree is further promoted by the nature of the communication that will be employed in the implementation,

where the CoCT uses SCOOT to monitor its TICs and update their timing plans from a central computer. By constructing a tree with a central entity at the root of the tree and the nodes making up the branches and leaves, optimal routes can automatically be established between the central entity and all the nodes of the network. This optimises the routes between the central SCOOT computer and the TICs of the CoCT.

The tree hierarchy accomplishes two tasks in a single step, firstly, it segments the entire network into different hierarchies, and secondly, it obtains optimum routes to all nodes in the network from the central station. There are products and routing protocols available that maintain mesh networks with optimal routes to all nodes in the network. They can, however, have high demands on bandwidth to be maintained and are often limited to relatively small networks. Due to the nature of the target implementation, where communication between a single central entity and multiple network nodes is required, it was decided to use the approach taken in TH(O)RP that only optimises routes between the central entity and the network nodes in an effort to minimise control traffic overhead and improve system scalability.

4.4 TH(O)RP operation

Because of the centralised point, namely the instation, it is chosen as the top of the hierarchy, or at the ROOT of the tree. A tree hierarchy is achieved by letting each node in the network choose a parent based on which node would provide it with the shortest route to the ROOT of the tree, where only a node that already has a valid parent can be selected by other nodes as their parent. In this manner, the tree hierarchy is naturally constructed around the instation. The resulting tree hierarchy will be similar to a binary tree, but it will have additional connections, reflecting the many connections between neighbouring nodes in mesh networks. In the event that a node loses connectivity to its current parent, one of these other connections can be used to connect to a new parent. An example network and the resulting tree hierarchy is displayed in Figure 4.1. By letting each node choose its parent as the next best hop based on a well defined link metric, the routing protocol can automatically route around problem areas in the network, such as areas with a lot of interference and poor links. This will result in an optimum and stable tree hierarchy with the instation at the ROOT of the tree. The link metric will promote more efficient routes by allowing the calculation of route costs, which nodes can use to select the ‘shortest path’ to the instation, and is discussed in Section 4.6.2.

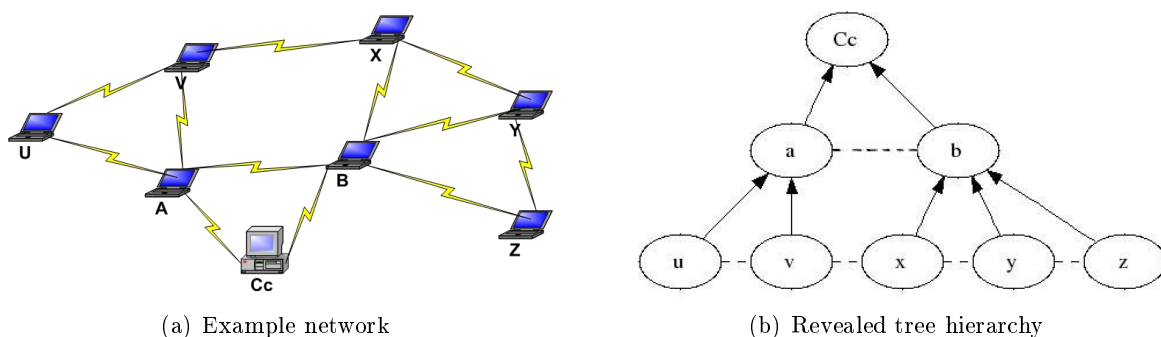


Figure 4.1: An example network and the revealed tree hierarchy

TH(O)RP uses the tree hierarchy as the structure along which routes are constructed, and message forwarding only take place up and down this hierarchy. Routes are established with a mechanism called the high-speed propagation path (HSPP) [29, 32] that allows routes to be pushed up the network hierarchy.

To illustrate HSPP, observe node X in Figure 4.2. When node X chooses node B as its parent, it pushes a route to itself up the hierarchy through node B. Upon reception of the route update, node B will enter a route to node X in its routing table and forward the message further up the hierarchy to node Cc. When node Cc receives the route update, it will make an entry in its routing table for node X and because it knows that it is the ROOT of the hierarchy, it will send a route reply down the hierarchy towards node X using the route that was just established. When node X receives the route reply, it makes an entry in its own routing table for node Cc. When this process is completed, node X knows how to reach node Cc and vice versa. If all the nodes in the network complete this process, the result is that all the nodes have a connection to the instation and the instation knows how to reach all nodes in the network.

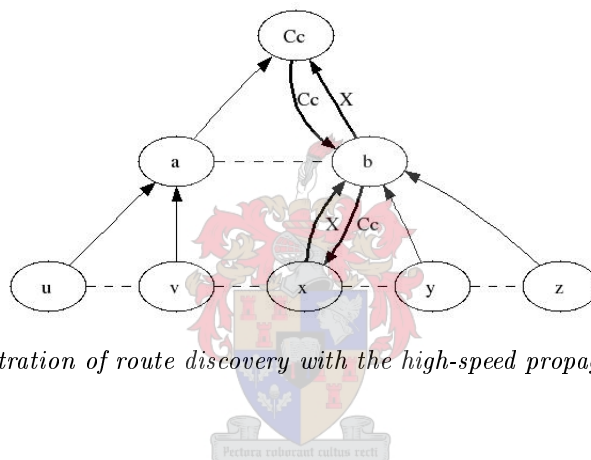


Figure 4.2: Illustration of route discovery with the high-speed propagation path (HSPP)

Although communication between the nodes of the network is not a requirement, this can easily be achieved. For instance, if node U from Figure 4.3 wants to establish a route to node X it only needs to push a route request to node X up the hierarchy. At each node higher up the hierarchy, the node checks whether it has a route to node X. If it has a route, it can send a route reply to node U specifying the cost of that route. If it does not have a route it will send the route request further up the hierarchy. If node X is in the network and has pushed a route to itself up the hierarchy, a route to node X will eventually be found. In Figure 4.3, this occurs when the route request reaches the instation (node Cc) at the top of the hierarchy. When the instation receives the route request to node X from node U, it realises that it has a valid route to node X and sends a route reply back to node U specifying the cost of the route to node X. Due to the nature of the tree hierarchy, node U will always find a route to node X without having to resort to network wide flooding. The only question is how far up the hierarchy it is required to search to find a route to node X.

The connection made between node U and node X is referred to as the first connection [29] and is not an optimum route. It can be optimised through localised flooding around the first connection. This is not implemented in the current version of TH(O)RP, and can be included in a future release.

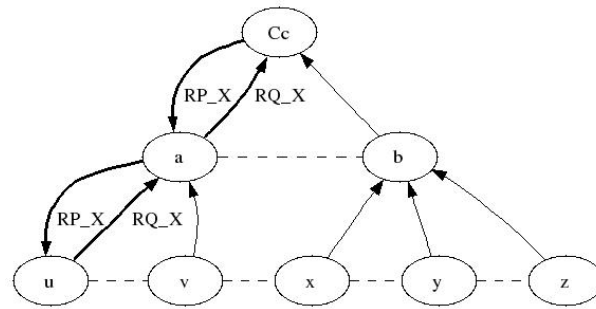


Figure 4.3: Illustration of route discovery between two nodes

Similar to the OLSR protocol [46, 26] that reduces the number of nodes that can forward a message, TH(O)RP also reduces the number of nodes that can forward messages to the nodes that have children. Figure 4.4 indicates eight nodes in the network. If all nodes are allowed to forward messages, all eight nodes will forward broadcast messages. By limiting the nodes that are allowed to forward messages to those that have children, it is reduced to three nodes, whilst obtaining the same results. In the case of an addressed message, a uni-cast message destined for a particular node, the effect of message forwarding is further reduced to only a subtree of the network containing the destination node.

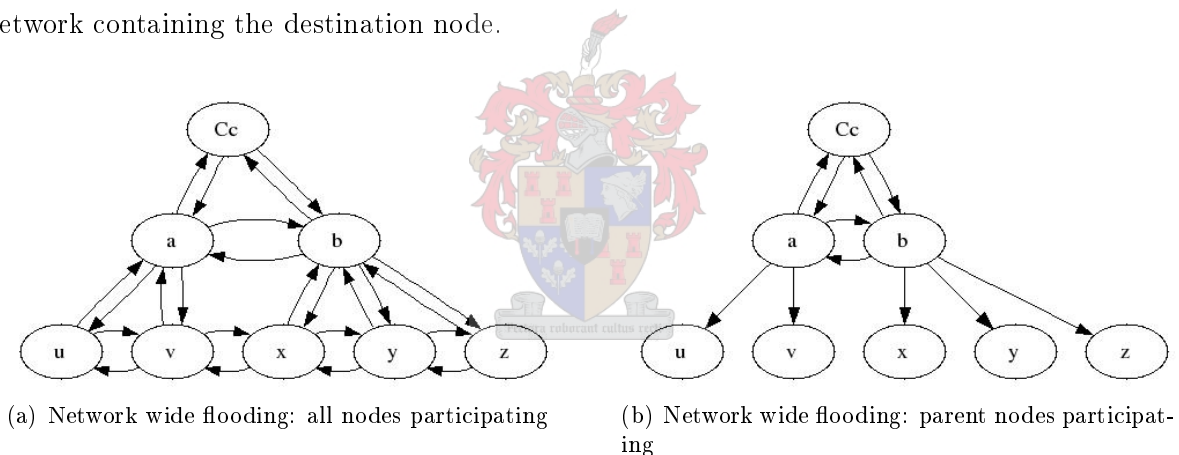


Figure 4.4: Illustration of reduced flooding

The functionality of TH(O)RP is divided into primary and secondary components. The primary components provide the base of the protocol and are crucial to its operation. The secondary components are supplementary and optimise the protocols' behaviour.

4.5 Primary components

In the previous section, the basic operation of the TH(O)RP protocol was discussed. In this section the various components of TH(O)RP are discussed. The protocol can be broken up into separate components, namely neighbour discovery and link sensing, parent selection, route discovery and maintenance, and message forwarding.

4.5.1 Neighbour discovery and link sensing

The neighbour discovery and link sensing components are responsible for gathering information about node neighbourhoods. A neighbourhood is defined as all the nodes that are in the immediate vicinity of a node, or nodes that are visible on the network interface of a node and that can be reached in a single hop. In the case of wireless networks, a neighbourhood is all the nodes within radio range.

Neighbour discovery

During neighbour discovery each node detects the neighbours in its vicinity. Neighbour discovery is achieved by generating periodic HELLO messages advertising neighbourhood information. The period of the HELLO messages can be adjusted to suit the nature of a network. For instance, static networks can employ longer periods between HELLO messages, whereas mobile, dynamic networks will require more frequent HELLO messages.

Each node in the network will periodically generate HELLO messages with information about its neighbourhood. In this manner a node can inform other nodes in its neighbourhood of its presence. When a node receives a HELLO message, it records the information of the node that issued the message in a table. When a node generates a HELLO message, it includes the information from this table, resulting in HELLO messages distributing the node's understanding of its neighbourhood. When a node receives a HELLO message that contains a neighbour entry for it, it knows that the node that issued the HELLO message is aware of its presence. In this manner all nodes become aware of the nodes in their respective neighbourhoods.

Link sensing

Link sensing [26] is the process through which a node obtains the state of the communication links in its neighbourhood. The possible states are asymmetric, symmetric and lost. The first two states describe one-way and two-way links respectively, with the final state describing a link that is no longer valid. With the aid of the information contained in the periodic HELLO messages, the state of links can be derived. See Figure 4.5 for an illustration of link sensing.

In Figure 4.5 node A initially issued a HELLO message with no neighbours included. When node B receives the HELLO message, it notes that node A is in its neighbourhood, but that it is an asymmetric link. Node B then issues a HELLO message with node A included as a neighbour with an asymmetric link type. Upon reception of this message, node A can record node B as a neighbour and derive that it has a symmetric link. When node A issues its next HELLO message, it includes node B as a neighbour with a symmetric link type and when node B receives this HELLO message, it can update its neighbour node A to have a symmetric link type. After the process finishes, both nodes A and B recorded each other as symmetric neighbours. In the event that the link between node A and node B is asymmetric, HELLO messages in one of the two directions would not be successfully transmitted and one node would register the other as asymmetric. Furthermore, one node would be unaware of the existence of the other.

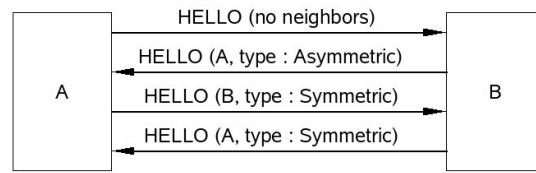


Figure 4.5: Calculating link state during link sensing [46]

4.5.2 Parent selection

The process of parent selection is one of the most important processes of TH(O)RP, as it determines the network hierarchy designating the communication routes. A parent node is chosen as the next best hop to the top of the hierarchy, or the instation. The net effect is that the resulting routes between the instation and the nodes of the network are optimum routes. A parent is selected from the nodes discovered in a neighbourhood. By periodically repeating the parent selection process, the tree hierarchy and resulting routes are continuously optimised. See Figure 4.6 for an illustration of the parent selection process.

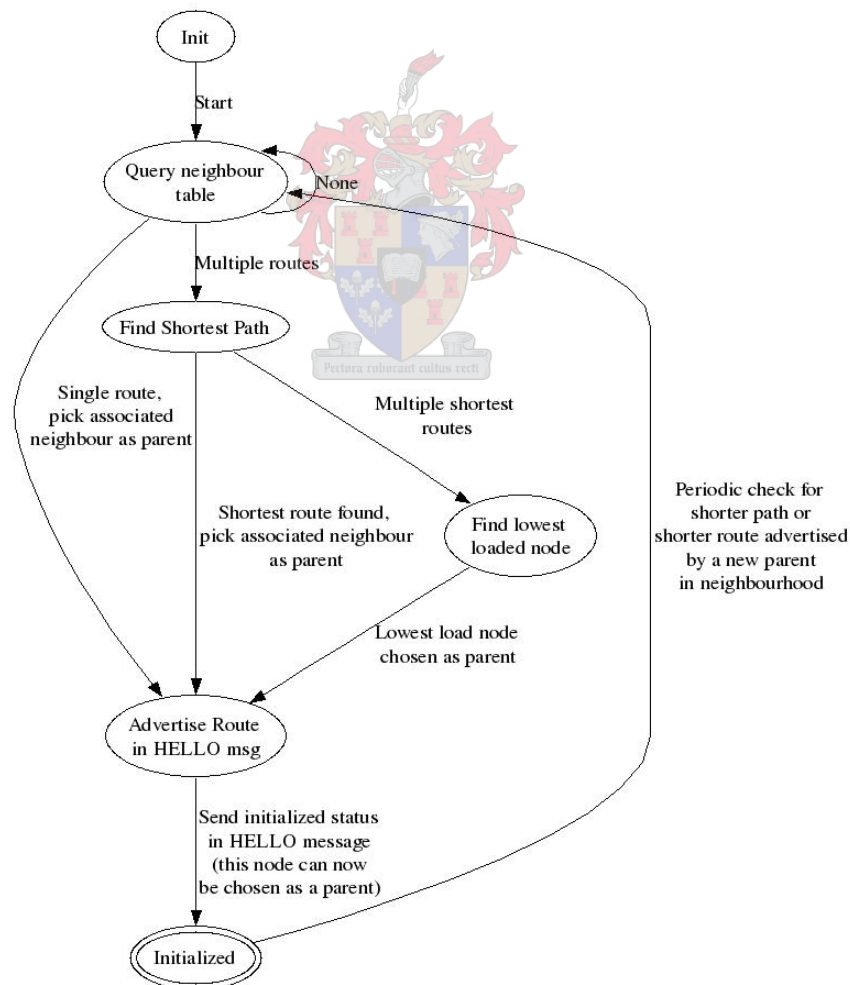


Figure 4.6: Illustration parent selection process

4.5.3 Route discovery and maintenance

The process of route discovery using HSPP was briefly discussed before. The routes between the instation and the nodes of the network are pro-actively maintained and ensure continuous connectivity. Every time a node selects a new parent, a new route is pushed up the hierarchy. This process is repeated for situations where a new or previous parent is selected to keep routes up to date.

Route discovery is achieved through route request (RREQ) and route reply (RREP) messages. When a node selects a new parent or updates its route, it issues a RREQ message. These RREQ messages are usually addressed to the node issuing the message, so the node is requesting a route for itself. A route can also be requested to any other node by issuing a RREQ message to the address of that node. When a node that has a route to the requested destination, receives the RREQ, it will generate a RREP. In the case of a self addressed RREQ, the RREQ will travel all the way up the hierarchy to the ROOT.

Routing information is stored in a distributed manner through the entire network hierarchy. No node ever knows the entire route hop for hop towards any destination. Each node only knows what the next hop is toward a destination. In this manner, the amount of routing information stored at a node is limited to a minimum. By utilising the collective knowledge of the network, a node can have a route to any destination in the network as long as it knows what the next hop towards that destination is.

4.5.4 Message forwarding

Messages received at route nodes, which are nodes selected as parents, will need to be forwarded to the message destination. In order to accomplish successful message forwarding, a default forwarding strategy is required. The default strategy will use the network hierarchy along with the distributed routing information to route messages. The forwarding strategy is as follows:

- HELLO and acknowledgement (ACK) messages, which only have a time to live (TTL) value of one, are not forwarded.
- Multi-hop control- and data traffic is recorded in a duplicate set. When a node receives such a message, it will check if it is already entered in the duplicate set. If it is entered and marked as previously processed or forwarded, the node will not process or forward the message again. This prevents messages from staying resident in a network long after they have already reached all intended recipients and prevents messages from being relayed back down the routes they came from. It also prevents count-to-infinity loops from occurring. There are addressed and broadcast messages, which are treated differently.
 - Addressed, multi-hop control- or data traffic like RREQ, RREP and DATA messages need to be forwarded in the direction of the destination. When a node receives such a message, it will check whether it has an entry for the destination in its routing table. If it has a route entry, it will forward the message towards the destination through the next hop of the route. If it does not have a route entry, it will forward the message higher up the tree hierarchy.
 - Broadcast, multi-hop control- and data traffic like MID, HNA and broadcast DATA messages, are not addressed to a specific node and need to be transmitted around the

entire network. When a node receives such a message, it will process the message and simply forward it again by broadcasting the message. The duplicate set will prevent messages from being relayed to where they came from.

- Whenever a packet is forwarded, the hop count parameter is incremented and the TTL parameter is decremented. This will avoid packets floating around the network for an indefinite period of time. When the TTL parameter reaches zero, the message is not forwarded again.

The forwarding strategy is summarised in Figure 4.7 and its purpose is to ensure that control- and data traffic are correctly forwarded to their required destinations. Via a plug-in interface, alternative forwarding strategies can later be implemented. This could be used to designate specialist nodes in the network with specialised forwarding strategies without having to define these strategies for all the nodes in the network.

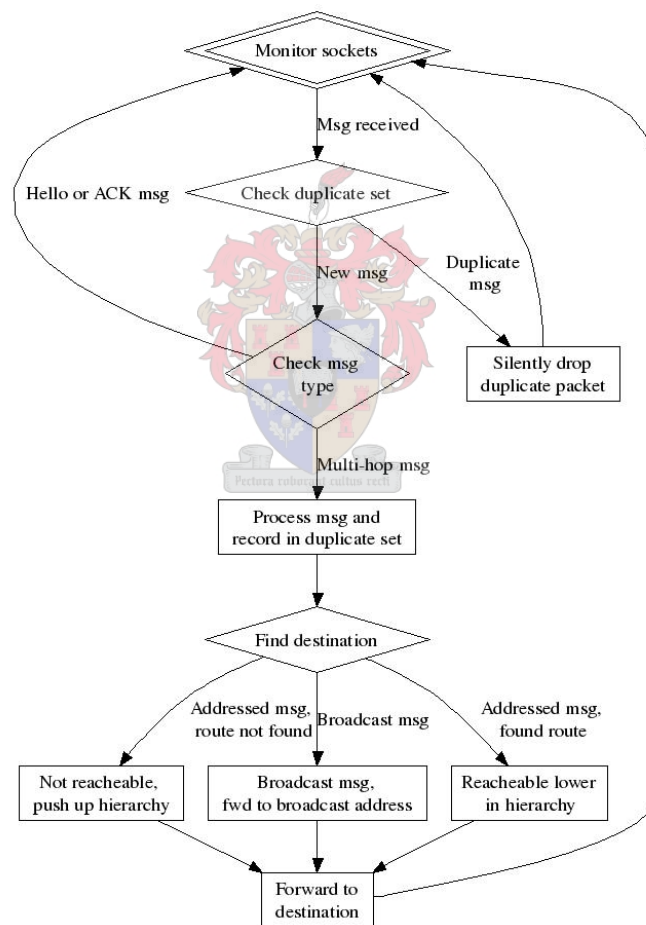


Figure 4.7: Illustration of default forwarding strategy

4.5.5 Unknown message handling

Because a plug-in interface is planned for TH(O)RP, there might be situations where some nodes run the original version of the protocol while others might have additional plug-ins defined, adding functionality and possibly new message types. This could cause certain messages to be unknown

to the nodes that do not contain these plug-ins. These unknown messages would simply be ignored by normal nodes as they would not know the format of these messages and where to obtain addressing information from them.

To enable message diversity in such networks it was decided that, if a node receives a message that it cannot interpret, it will simply record the message in its duplicate set and forward the message to the broadcast address. Because the source and destination addresses of unknown messages cannot be derived, they need to be sent around the entire network or until a node able to interpret the message, receives it. The duplicate set will prevent the message from traversing down routes that it came from and will allow it to enter parts of the network not yet reached. It should once again be noted that only nodes that have children are allowed to forward messages.

Although these messages might be of an unknown type, they have to be valid TH(O)RP messages, containing valid message headers. Just the body of these messages may be unknown.

4.6 Secondary components

Some secondary components were included to improve the normal operation of TH(O)RP. They are supplementary and do not effect its basic operation. These components do however, provide valuable optimisations to the basic implementation of the protocol.

4.6.1 Link hysteresis

Because of the dynamic characteristics of any wireless environment, messages are often lost, meaning that the state of links could oscillate, causing the routes calculated by TH(O)RP to oscillate. This would have a negative effect on the perceived network stability and would trick the protocol into making numerous changes to routes, resulting in a continually changing tree hierarchy.

In order to avoid the oscillation of link states, link hysteresis [46, 26] is used. Link hysteresis causes a delay in changes to link states, resulting in a state change only after a certain amount of losses are registered. Link hysteresis also makes TH(O)RP more robust against short lived connections, as it will avoid selecting nodes as parents that are only visible for a short period of time. Link hysteresis is illustrated in Figure 4.8. At time A the link state is upgraded to symmetric and at time B the state is downgraded to asymmetric. The separation between the upper and lower thresholds act as a buffer and a link really needs to improve or deteriorate for the link state to be effected. When calculating the link state with link hysteresis, the stability- and instability rules from [26] are used:

The stability rule

$$L_{new_link_quality} = (1 - H_{scaling}) * L_{old_link_quality} + H_{scaling} \quad (4.6.1)$$

The instability rule

$$L_{new_link_quality} = (1 - H_{scaling}) * L_{old_link_quality} \quad (4.6.2)$$

where $H_{scaling} = 0.5$, and from Figure 4.8 the upper and lower thresholds are 0.8 and 0.3 respectively.

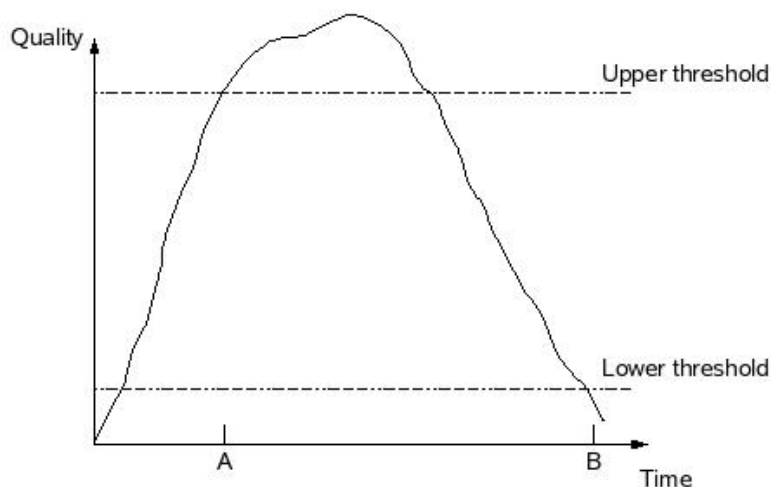


Figure 4.8: *Illustration of link hysteresis*

The stability rule is calculated every time that a valid TH(O)RP message is received. Packet losses are tracked using TH(O)RP packet sequence numbers and the hold time parameter of HELLO messages. Thus, if a HELLO message fails to arrive before its hold time expires, a packet loss is registered and the instability rule is executed. The suggested parameters for link hysteresis from [26] were used, but the behaviour of link hysteresis can be finely tuned via these parameters to suit a variety of different networking environments.

4.6.2 Link metric

In normal shortest path protocols, the path with the least amount of hops is determined to be the optimal route. In wireless networks this is often not the case. Factors such as loss rate and bandwidth contribute to the quality of a link and subsequently the route. By including these factors into calculating optimum routes, faster and more stable routes between nodes can be obtained.

In [28, 31, 34, 30] a link metric is proposed that attempts to take the impairment of wireless channels into account when calculating the ‘shortest path’ between nodes. The link metric is based on the expected transmission time (ETT) of packets over a link. Each link can then be weighed with an ETT value, and by combining all the ETT metrics for each hop along a route the weighted cumulative ETT (WCETT) metric for that route can be obtained. By using WCETTs, the ‘shortest paths’ through the network can be determined.

In order to obtain the ETT of a link, the expected transmission cost (ETX) of that link must be calculated. The ETX metric measures the expected number of transmissions, including retransmissions, needed to send a packet across a link. To calculate the ETX metric, the underlying packet loss probability of the communication link, in both the forward and reverse directions (p_f and p_r respectively), need to be measured. The expected number of transmissions [31] for successful communication can then be calculated. Let us assume, that for a successful transmission to occur, a packet must also be successfully acknowledged. Let p denote the probability that packet transmission between two nodes is not successful [31]:

$$p = 1 - (1 - p_f) * (1 - p_r) \quad (4.6.3)$$

The probability that a packet will be successfully delivered after k attempts, denoted $s(k)$ [31] is:

$$s(k) = p^{k-1} * (1 - p) \quad (4.6.4)$$

The expected number of retransmissions required to successfully deliver a packet between two nodes can be calculated as per [31]:

$$ETX = \sum_{k=1}^{\infty} k * s(k) = \frac{1}{1 - p} = \frac{1}{(1 - p_f)(1 - p_r)} \quad (4.6.5)$$

The ETT of a link is also defined as the bandwidth adjusted ETX. Thus, the ETX (the number of expected retransmissions) is multiplied by the link bandwidth to obtain the time spent transmitting a packet. Let S denote the size of the packet and B the bandwidth of the link. Then [31]:

$$ETT = ETX * \frac{S}{B} \quad (4.6.6)$$

In this form, the back-off time spent waiting for the radio channel is not incorporated. In [31] Appendix A, there is an alternative definition that includes the back off time. With ETT weights defined, the WCETT metric for a n-hop route can be calculated as [31]:¹

$$WCETT = \sum_{i=1}^n ETT_i \quad (4.6.7)$$

In the current version of TH(O)RP, only the ETX and WCETX metrics are implemented, but in future versions the ETT and WCETT metrics might be incorporated, as the deployment of TH(O)RP is planned for network environments with multiple network interfaces with varying bandwidths.

While these metrics usually work well for single radio nodes, multiple radio nodes might exist and the metric will need to be modified to take multiple radios into account. This is further discussed in [31].

By using a well defined link metric, TH(O)RP favours stable links and automatically finds routes around poor, lossy areas in the network. However, if the net effect of a route around a bad spot incurs a higher cost than a route through that spot, the protocol will choose the route through the bad spot as it has a lower route cost.

In a comparison between link quality metrics for ad hoc wireless routing, it was found in [30] that the ETX metric outperformed other metrics when used in static network environments with stationary nodes. In dynamic networks with fast changing topologies, it was found that minimum hop-count routing performed best as it was better able to adapt to fast changing topologies. In the envisioned communication network for the CoCT, nodes will have static placements with possible link failures. Thus, the ETX and WCETX routing metrics will provide the most efficient routing.

¹This version of the WCETT metric does not take channel diversity into account, see [31, 34] for a version with channel diversity included.

Calculation of link metric in TH(O)RP

In the current implementation of TH(O)RP only the ETX metric and WCETX weights are used for links and routes respectively. The WCETX weights for a n-hop route is calculated as:

$$WCETX = \sum_{i=1}^n ETX_i \quad (4.6.8)$$

In TH(O)RP, the ETX metric is calculated by calculating the average number of messages correctly received from a neighbour. This is achieved by looking at the last N messages, where N denotes a window size, that were expected and counting the number of messages registered as lost. The probability that a message is successfully transmitted in the forward direction is then:

$$1 - p_f = \frac{(Total - Lost)}{Window\ size} \quad (4.6.9)$$

where Total = number of messages received (maximum of N)
 Lost = number of messages lost (maximum of N)
 Window size = size of window (N)

The reverse probability is the forward probabilities neighbour nodes and are obtained through HELLO messages from the neighbourhood. The route costs are calculated when a node selects a parent and pushes a route to itself up the hierarchy, using the HSPP. At each hop along the route, the ETX metric is added to a parameter in the route request message, named WCETX, which summates the individual ETX metrics of hops along a route.

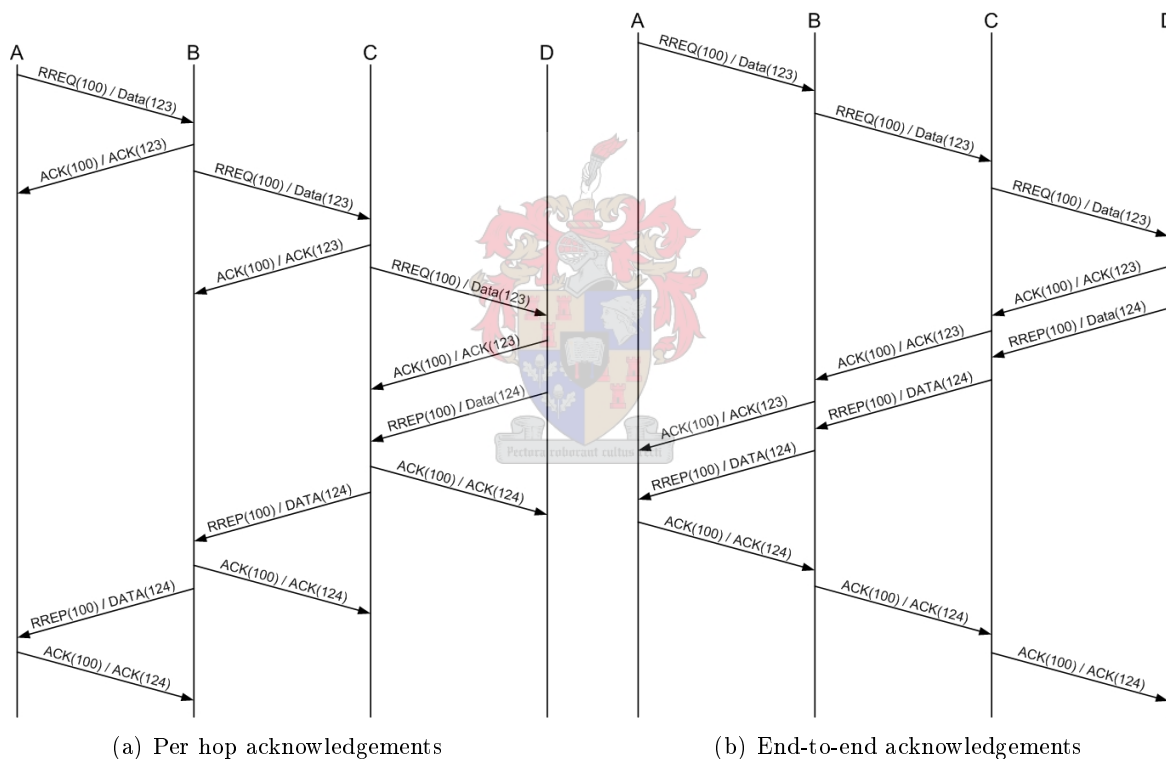
4.6.3 Message delivery and indirect acknowledgements

In order to make the message delivery of TH(O)RP more reliable and promote network stability, a basic delivery system was implemented. It allows multiple retransmissions per message per hop. A message acknowledgement (ACK) system was also required, that could inform the protocol of successful message delivery. When a node successfully receives a message, it issues an ACK message to inform the sending node that it has successfully received the message. Many different types of ACK methods exist. An ACK can be sent for each message received, or an ACK can be sent to acknowledge a sequence of messages, known as windowed ACKs. Windowed ACKs can use positive ACKs to acknowledge the last successfully received message based on a window, or negative ACKs (NACK) that specify message that have not yet arrived. The ACK system used in TH(O)RP is positive ACKs acknowledging every message received. It should be noted that the system employed here is unrelated to any transport layer protocols that might be used, and it only applies to multi hop messages, thus excluding HELLO messages.

When communication occurs over multiple hops, it has to be decided when an ACK is to be issued. Firstly, it can be issued at each hop along a route. Secondly, an ACK can be issued at each end of the route, thus a message can travel all the way through a network to a destination, with the destination issuing the ACK. The advantages and disadvantages are summarised in Table 4.1 and the two methods are illustrated in Figure 4.9.

Table 4.1: Comparison of two types of acknowledgements

Type	Advantages	Disadvantages
Per hop	Ensured that a message successfully arrives at intermediate hops. If a message is lost half way through a route, it can be retransmitted by intermediate nodes along the route. Provides fast response to message losses.	Increased control traffic overhead because each hop issues ACK messages. Increased management by intermediate nodes, because they need to keep track of transmitted messages.
End-to-end	Intermediate nodes are not required to keep track of transmitted messages.	Lacks robustness due to the fact that lost messages must be retransmitted from the start of route, even though it might have progressed through most of the route. Suffers from a slow response to message losses.

**Figure 4.9:** Comparison of two acknowledgement techniques

It was decided to use a hybrid of the per-hop acknowledgements since end-to-end acknowledgements lack robustness. The idea for the hybrid implementation was found in [37] which introduces the concept of multiple access with reduced handshake (MARCH). Because all the traffic generated by TH(O)RP is broadcast over UDP sockets, all nodes can intercept messages that are being transmitted within their neighbourhoods. The result is that when a node forwards a message, the node that issued the original message can see that the message was forwarded by the next hop on the route to the destination. Because the forwarding node already indirectly notifies the source node that it has received the message, the forwarded message can be used as an indirect acknowledgement, therefore the use of the term indirect ACK. When a node is

not required to forward a message, it sends a direct ACK. See Figure 4.10 for an illustration of indirect acknowledgements. By using indirect acknowledgements, the same durability is obtained as that of per-hop acknowledgements, without the increase in control traffic.

When comparing the different ACK methods, it can be seen from Figures 4.9 and 4.10, that a minimum of two ACKs is used for indirect ACK, compared to six ACKs for the other two methods. It was stated that these are the minimums, because all messages could potentially get lost or corrupted during transmission, so the number of ACK messages for both instances could increase.

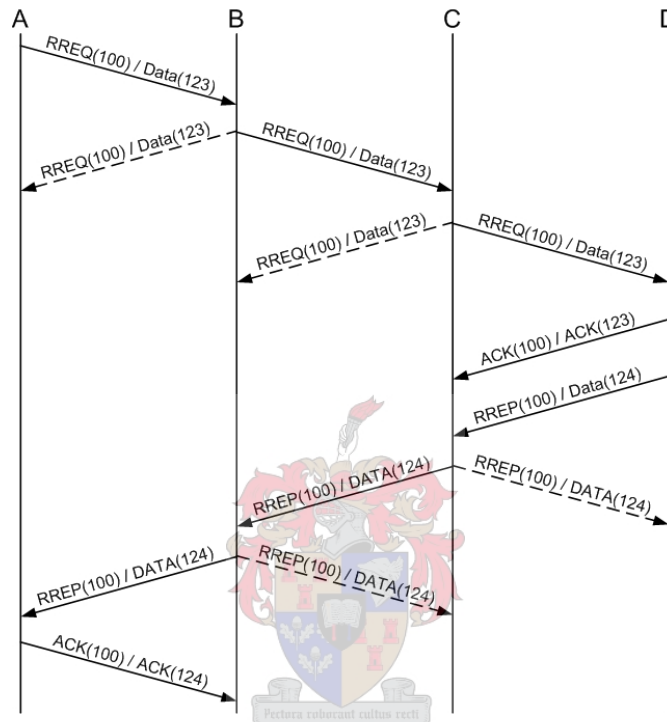


Figure 4.10: Indirect acknowledgements

The delivery system employed in TH(O)RP should not be confused with the transport layer. It only serves to make the protocol more robust to network losses in the network layer, providing more reliable delivery of control and data information. In order to ensure guaranteed delivery of application layer information, a transport layer delivery system should be used.

4.7 Packet formats

The TH(O)RP packets are divided into two groups. The first is control traffic that is used to maintain routes and information about the network topology. The second is data traffic, which transports application data around the network. The message types employed by TH(O)RP are HELLO, RREQ, RREP, DATA and ACK messages.

A TH(O)RP packet can consist of multiple TH(O)RP messages. This allows piggybacking of messages together in the same packet, and reduces the traffic in the network. The generic TH(O)RP packet format is displayed in Figure 4.11. The TH(O)RP message formats can be seen in Appendix A.

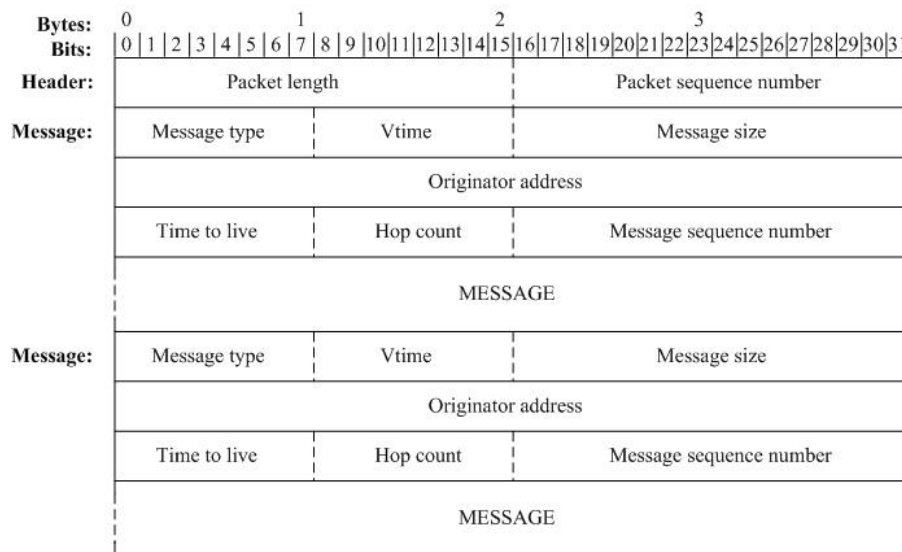


Figure 4.11: The Generic TH(O)RP packet

4.8 Modular protocol design

The TH(O)RP routing protocol follows the same modular design concept as found in [46]. A modular design logically segments the protocol into individual components, which can be designed and developed separately. Once the development of a module is finished, it can be combined with the other modules to make a fully functional protocol. With a modular approach, improvements and optimisations can be applied to individual modules without compromising the operation of the entire protocol. The modules are:

- Socket parser
- Packet parser
- Information repositories
- Scheduler
- Plug-in interface

See Figures 4.12 and 4.13 for the interaction between the protocol parts.

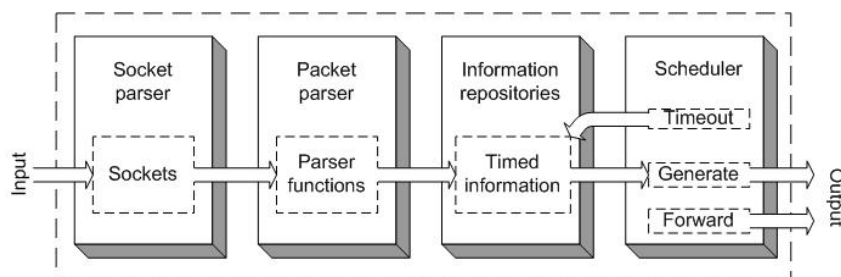


Figure 4.12: Modular design of TH(O)RP

4.8.1 Socket parser

The socket parser looks for incoming TH(O)RP traffic and will call associated functions when messages are received. The current implementation of the protocol has a single UDP socket defined through which both control and data traffic are sent and received. Provisions will be made for multiple sockets and in future implementations separate sockets might be defined, such as a UDP socket for control traffic and a TCP variant socket that is suited to wireless mesh networks for data traffic. Additional sockets can be defined to communicate with plug-ins which are also monitored by the socket parser.

4.8.2 Packet parser

The packet parser will receive TH(O)RP packets from the socket parser along with the associated function it should execute. If it receives a control packet, the packet parser will know that it must perform a control operation. Before the packet parser executes any commands, it needs to verify that the packet was destined for the current node. If the packet was not destined for a node, it is either forwarded or silently dropped. The packet parser processes received packets and updates the information repositories based on the packets received.

4.8.3 Information repositories

TH(O)RP is a table driven protocol with pro-actively maintained routes. Because of the dynamic nature of wireless networks, most of the information stored in the repositories will be timed and will expire unless they are periodically refreshed or updated. This is in order to prevent stale table entries that no longer reflect the state of the network topology. Information about one hop neighbours, link states and qualities, routes, duplicate messages and re-transmittable messages will be kept. The repositories kept are a neighbour table, link set, route table, duplicate set and retry table. Future tables could include a MID set and HNA set which will be briefly discussed later.

Neighbour table

The neighbour table contains information about the one hop neighbourhood of TH(O)RP nodes. The information includes the node addresses, whether they are child or parent nodes or whether they are symmetric or asymmetric neighbours. It also contains information about whether a neighbour has a valid parent entry, how many children it has and the cost of its route to the top of the hierarchy. Parent selection uses the neighbour table to identify viable parent nodes in a neighbourhood.

Link set

The link set contains information about the links between a node and its neighbours. It contains the link state and link qualities. It also records how many messages were lost or successfully received between a node and a particular neighbour, which is used to calculate the link qualities.

Route table

The route table contains all routes known by a node. The route table will include entries for the instation at the ROOT of the tree hierarchy and any routes to nodes below a node in the hierarchy. The route table keeps information on the next hop to any destination, the number of hops required to get there and the cost, in WCETX metric, to the destination. When nodes have multiple interfaces, certain nodes might only be reachable over certain network interfaces, so the route table also records the name of the network interface over which a node is reachable.

Duplicate set

The duplicate set records the sequence number of messages previously seen by a node. It also specifies whether the message was previously processed or forwarded and prevents a node from re-processing messages. The duplicate set holds a message for a certain time period to prevent messages that exist in the network for long time periods from being reprocessed.

Retry table

The retry table contains forwarded messages that might require retransmission. If a message is successfully acknowledged by a next hop or destination node, it is removed from the retry table. If the message or the associated ACK was lost, a timer expires and the message is retransmitted. After multiple retransmission attempts without being successfully acknowledged the message is removed from the retry table and dropped.

4.8.4 Scheduler

The event scheduler is responsible for scheduling all events in TH(O)RP. The events include message generation- and table time-out events. Events can be dynamically added or removed from the events registered with the scheduler, so plug-ins of the protocol can add or remove events. This provides additional extensibility to the protocol.

4.8.5 Plug-in interface

A well defined plug-in interface is used in [46], which allows the implementation of the OLSR routing protocol developed in [46] to be expanded by plug-ins without modifying the core of the protocol. This provides tremendous flexibility and extensibility to that protocol. It was decided to include a similar plug-in interface in TH(O)RP. The plug-in interface provides a well defined interface to interact with almost all the components of the protocol. A good example plug-in, is a security plug-in that would provide network security to the protocol as the basic implementation includes no security measures. See Figure 4.13 for an example of how a security plug-in might work. Such a security plug-in might be used to encrypt or decrypt TH(O)RP packets or to verify sender and message authenticity with certificates of message checksums.

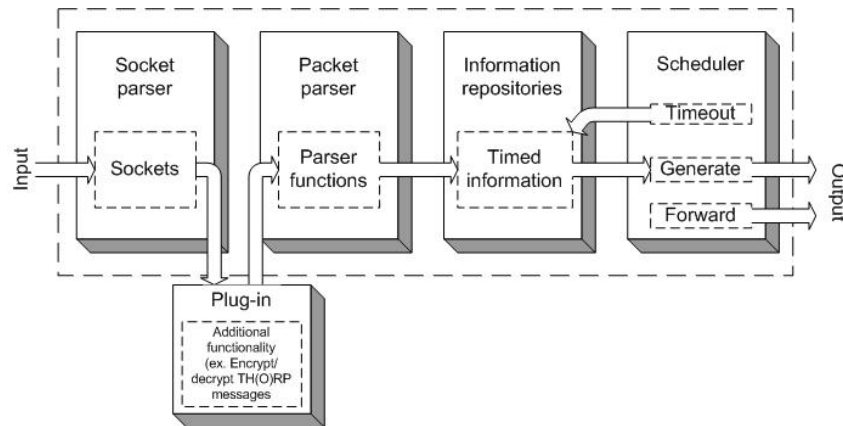


Figure 4.13: Example of a security plug-in

4.9 Protocol development cycle

TH(O)RP was developed in three stages, namely the design stage, simulation stage and software implementation stage. In this chapter, the components of the protocol were designed. In Chapters 6 and 7 the protocol is integrated into the OMNET++ simulation environment and tested. The necessary modifications and optimisations were applied. When the results from the simulation were satisfactory, the protocol was converted to C. The C working version is described in Chapter 8. See Figure 4.14 for the full development cycle. Any future developments made to the protocol will also follow this development cycle.

4.10 Future functionality

The core functionality of TH(O)RP has been implemented, but some additional functionality can be added in the future to extend its operation. These are complementary functions and can be used to optimise the protocols' behaviour, or enable additional services to run on top of the protocol.

4.10.1 Multiple interface declaration

Because the target network implementation proposed in this project has a mixed technology architecture, TH(O)RP nodes might have multiple network interfaces. They can therefore be multi-homed and can be reached over multiple network interfaces. To uniquely identify a node in a network over a certain network interface, a node will be required to have multiple network addresses, or aliases. Therefore, the protocol has a main address and an alias for each additional network interface.

To distribute the aliases of nodes around the network a multiple interface declaration (MID) [26] message could be used. This will distribute a node's main address and its aliases around the network, and enable other nodes to discern from an aliased addresses which node it is communicating with.

The current software implementations of TH(O)RP generate MID messages, but this functionality has yet to be fully incorporated.

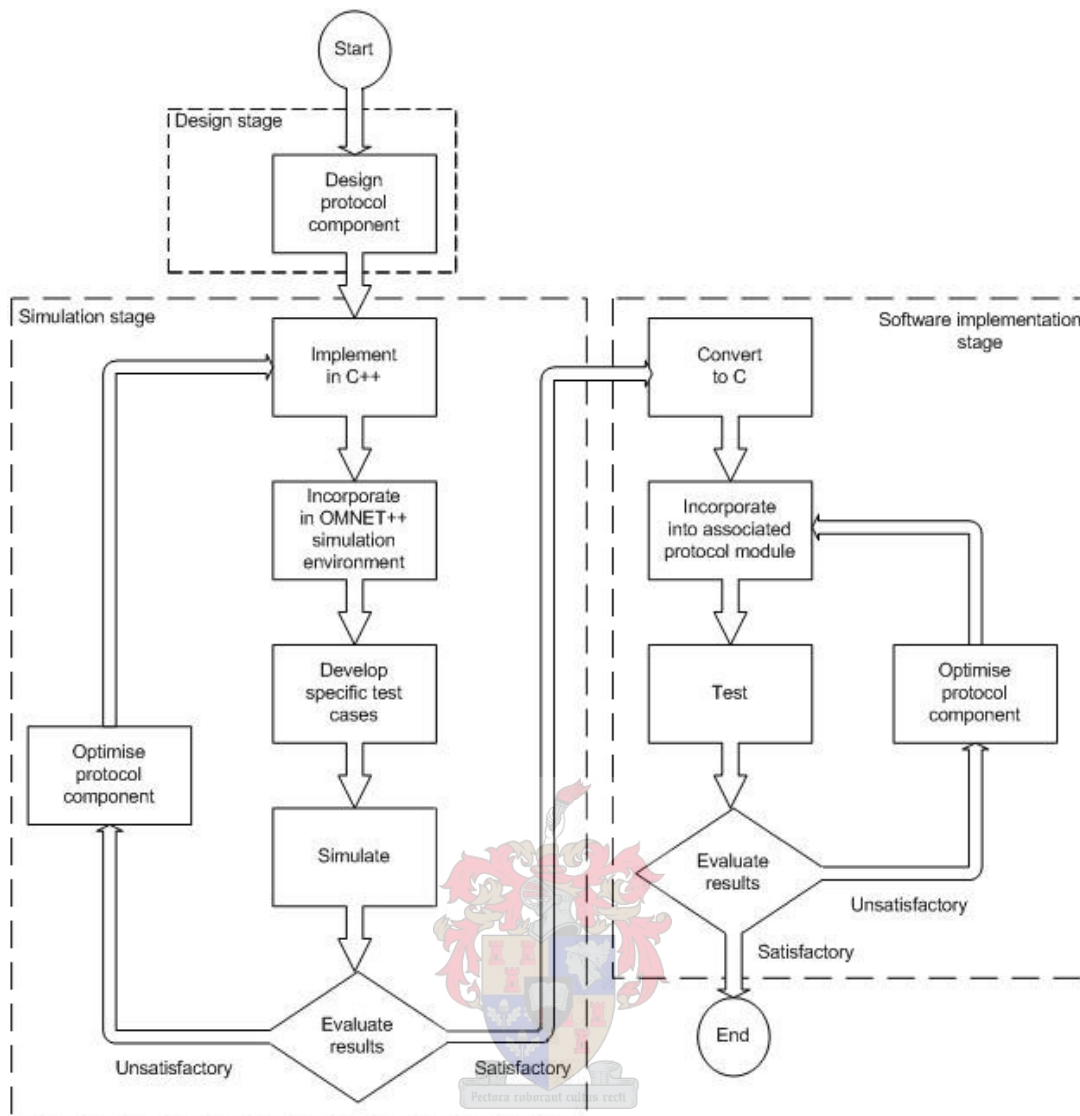


Figure 4.14: The protocol development cycle

4.10.2 Internet connectivity via host and network association

In future versions, TH(O)RP nodes would be able to connect to the Internet via gateway nodes. Knowledge of such gateways will, however, need to be distributed around the network, so that nodes are aware of Internet gateways and where they could be reached. A method to distribute information about gateway nodes around a network is with host and network association (HNA) [26, 23, 22] messages. A gateway node can announce its connectivity to other nodes in the network by flooding HNA messages throughout the network.

The current software implementations of TH(O)RP generate HNA messages, but this functionality has yet to be fully incorporated.

4.10.3 Security

It is well known that wireless communication can have weak security. This stresses the need to secure TH(O)RP, especially in view of its intended implementation. By implementing a plug-in that adds security checks to the messages being transmitted, a level of security could be added.

4.10.4 Better hierarchy structure via node willingness

Although TH(O)RP employs a link metric that should prioritise nodes with good links as candidates for parent selection, in certain situations network devices might be battery powered, or certain nodes need to be specifically prioritised as candidates. This could be achieved by specifying a node's willingness to be chosen as a parent, effectively pushing it higher or lower in the tree hierarchy. For instance, WiMAX nodes could be given a high willingness, while battery powered devices could be given a low willingness. The current HELLO messages of the protocol has a field to employ node willingness, but this has not yet been fully implemented.

4.10.5 Quality of Service

When additional services other than ITS related services are employed over TH(O)RP, it would be important to provide a quality of service (QoS) guarantee to time critical services, such as the TRAFX protocol. This would allow certain services to be prioritised over others and could be used to efficiently manage the bandwidth of the communications network.

4.11 Conclusion

TH(O)RP was designed with the primary goal of delivering a stable, scalable network environment with low latencies. The tree hierarchy of the protocol effectively segments the network into smaller parts and allows for better scalability of the routing protocol. The hysteresis and link quality metrics employed promote a more stable network hierarchy. The latency of the protocol is theoretically defined in Chapter 5 and is tested with simulations in Chapter 7.

The protocol manually defines the instation as the top of the tree hierarchy. This approach could result in tree hierarchies that are poorly balanced, as it manually specifies the top of the hierarchy, which might not be the central point of the network. This approach does, however, optimise the routes between the instation and the outstations of the network, which is desirable.

TH(O)RP combines components from other routing protocols, particularly the OLSR and OrderOne Networks protocols, to achieve a protocol well suited to facilitate communication of an ITS system using SCOOT. In addition to this, it adds further functionality to improve its operation in an ITS implementation, namely:

- Provides more efficient routing through the use of the ETX link- and WCETX route metric.
- Improved protocol scaling through use of tree hierarchy.
- Improved tree hierarchy stability with reliable message delivery in the network layer.
- Adds an efficient network layer acknowledgement mechanism for multi-hop networks.
- Optimises routes between the network nodes and a central entity, for example the TICs and SCOOT computer of the CoCT.

Chapter 5

Networking latency

In order to estimate the response and throughput of the different communications technologies, the latency of the network needs to be modeled. In order to model network latency, the network characteristics have to be taken into account.

In this chapter, latency models are developed, that can be used to estimate the performance of a network with different communication technologies. The models will also be able to indicate the bandwidth requirement to effectively process data generated by nodes of the network.

5.1 Terms and Definitions

5.1.1 Bandwidth

Bandwidth is the transmission capacity of an electronic pathway such as a communications line, computer bus or computer channel. In a digital line, it is measured in bits per second (bit/s) or bytes per second (byte/s). In an analogue or digital signal embedded in a carrier, bandwidth is the difference between the highest and lowest frequencies and is measured in Hertz (kHz, MHz, GHz) [4]. Bandwidth is related to the rate at which data can be transferred, either in bits/s (b/s) or bytes/s (B/s).

5.1.2 Latency

Latency is the amount of time it takes for a message to traverse a system. In a computer network, it is an expression of how much time it takes for a packet of data to get from one designated point to another. It is sometimes measured as the time required for a packet to be returned to its sender. This can also be seen as the response time, or round-trip-time (RTT) of the network.

Latency depends on the speed of the transmission medium used by a network, namely copper wire, fibre optic cables or radio waves, and the delays in the transmission by devices in the network, for example routers and modems. A low latency indicates a high network throughput.

Latency and throughput are the two most fundamental measures of network performance. They are closely related, but where latency measures the amount of time between the start of an action and its completion, throughput is the total volume of data moved in a given amount of time.

5.2 Sources of latency

There are two main groups for sources of latency, namely external and internal sources. The external sources have to do with the actual time needed to transmit data over a communication medium, for example the speed of light, distance, and noise levels in the medium. Some of the internal sources are processing delays, queueing delays, search delays, wait- and back-off times. Another source of latency is the control traffic that is required to maintain a network and transmit data across the network.

5.3 Assumptions for latency models

This chapter develops models for estimating the latency of a network. In order to simplify the working models, a few assumptions are made:

- all communication technologies used can be configured in a multi-hop network architecture.
- no collisions with other users (dedicated channel).
- all internal sources of latency are small enough to ignore.
- propagation delay, $T = \frac{\text{Distance}}{c}$, where c is the speed of light, is small enough to ignore.
- if a packet is lost, it is retransmitted immediately.
- have channels with a 10% drop rate (for the retransmission models only)

5.4 Modeling latency with queueing theory

To effectively estimate what the latency for each communication technology is, latency models are developed. The models are developed using basic queueing theory [49], and they use the device parameters of the different communication technologies from Section 2.2, as input parameters.

Although a model cannot be set up to cover every conceivable scenario that could occur in a multi-hop network, models are set up for a few general cases. These include one hop and multi-hop models. The one hop models include cases with one-to-one, one-to-many and many-to-one communication. The multi-hop models include cases with one-to-one-to-one and many-to-many-to-one communication. Finally, a case where messages have to be retransmitted due to channel noise, which causes data to be corrupted or lost, is included.

5.4.1 The basic latency model

Because a dedicated channel is assumed, the latency of a communication network can be calculated if the structure of the data to be transmitted, and how it is packaged is known. In other words, if the amount of control overhead required to transmit a certain amount of data is known, the time it would take to transmit those bits of data can be calculated. In Figure 5.1 latency is defined as the amount of time that expires from when data is transmitted (at node A) to when that data arrives at the destination (at node B).

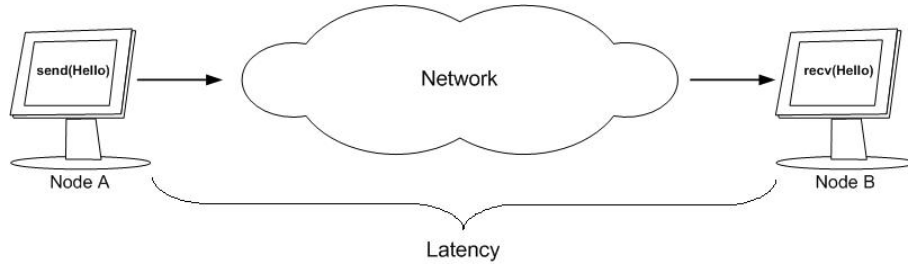


Figure 5.1: Basic latency model

The time required to transmit D bits, is the time required to transmit a single bit multiplied by the number of bits to transmit. The latency models use the device bandwidth of a communication technology, to calculate the time required to transmit a bit. The time required to transmit a packet of D bits is:

$$t_{data} = t_{bit} \cdot \mu_d \quad (5.4.1)$$

where μ_d is D bits and t_{bit} is the transfer time per bit. The models also consider the MAC and routing protocols used in the network as they introduce control traffic to successfully transmit data to a recipient. The control traffic overhead can be due to multiple control messages, for example, request to send (RTS), clear to send (CTS) and acknowledgement (ACK) messages, as well as the headers of data packets. Control traffic varies for different protocols, and the control traffic messages are analysed in Section 5.5.1. With $t_{control}$ defined as the time it takes to transmit control traffic, the total time required to transmit a packet is calculated as:

$$t_{packet} = t_{data} + t_{control} \quad (5.4.2)$$

All transmission networks are susceptible to errors that occur due to noise in the communication channel, potentially causing data packets to be corrupted or lost. This in turn, causes some packets to be retransmitted, which incurs an additional cost to the latency of the system¹. The retry time can be defined as the probability that an error will occur (p_{err}), multiplied by the time to transmit a packet:

$$t_{retry} = p_{err} \cdot t_{packet} \quad (5.4.3)$$

The total time to transmit a packet in the presence of noise is:

$$t_{total} = \Sigma t = t_{packet} + t_{retry} = (1 + p_{err}) \cdot t_{packet} \quad (5.4.4)$$

The service rate, μ , of the model is [49]:

$$\mu = \frac{1}{t_{total}} \quad (5.4.5)$$

The arrival rate, λ , is the rate at which messages are generated.

¹This does not include retransmissions due to collisions with other clients as the models assume a dedicated channel.

The traffic intensity is defined as [49]:

$$\rho = \frac{\lambda}{\mu} = \begin{cases} < 1 & \text{stable} \\ \geq 1 & \text{unstable (overflow)} \end{cases}$$

The queue length, or the number of events in the system is given as [49]:

$$N = \frac{\rho}{1 - \rho} \tag{5.4.6}$$

The latency of the system is calculated by [49]:

$$T_{wait} = \frac{N}{\lambda} = \frac{1}{\mu - \lambda} \tag{5.4.7}$$

This is known as Little’s result [49] and includes all service times, which is the transmission and processing latency of the system. The basic model is further refined to include retransmissions and multiple destinations by making different substitutions for the λ and μ parameters. This is discussed in Section 5.5.1.

5.4.2 One hop latency models

A few basic one hop models to model a single hop between nodes in the communications network are defined here. These models can be seen in Figure 5.2.

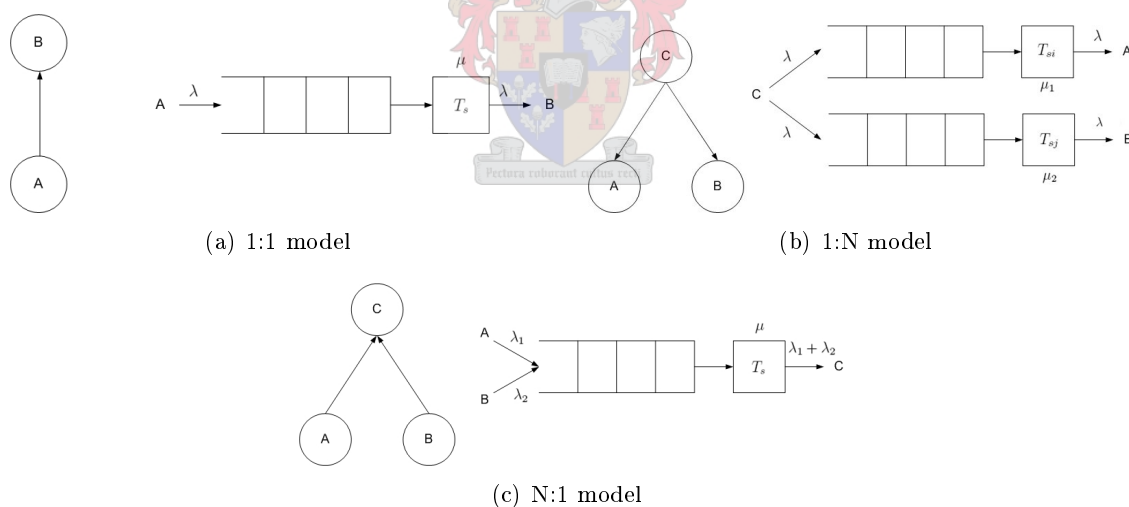


Figure 5.2: One hop queueing models

5.4.3 Multiple hop latency models

Here more complex models with multiple source and destination nodes and multiple hops are defined. These models can be seen in Figure 5.3.

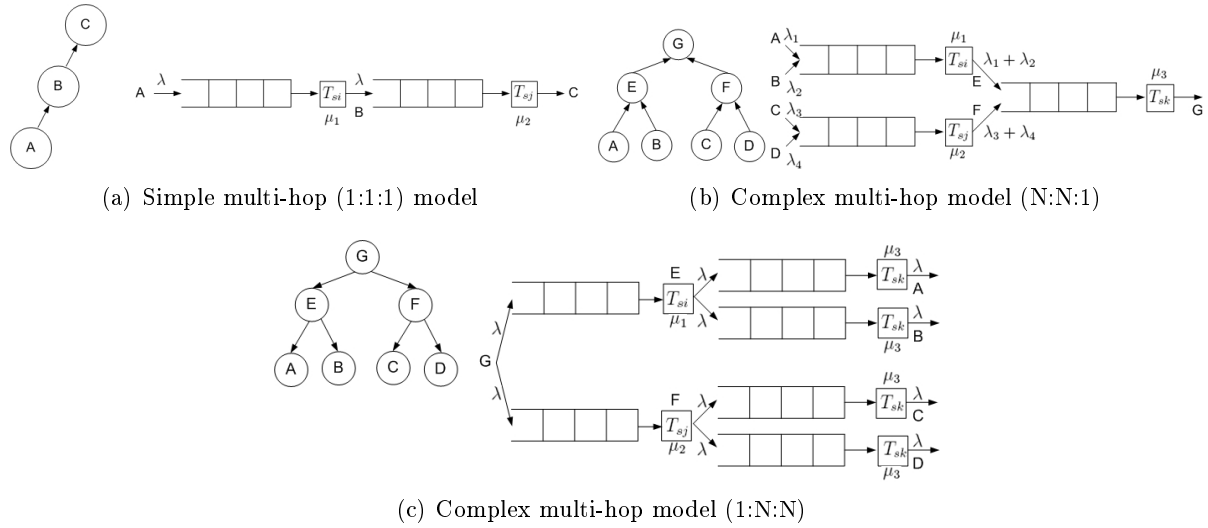


Figure 5.3: Multi-hop queueing models

5.4.4 Latency models with retransmissions due to noise

In any communication network, additive noise and interference can corrupt data. In wired networks this could be electromagnetic interference or cross-talk between cables that run parallel to each other. In wireless networks this could be due to radio interference, microwaves, or other radios using the same frequency band.

This means that for every bit of data transmitted, there is a probability that it might be lost and would require retransmission, which increases the amount of data that needs to be processed. The retransmission model is similar to the 1 to 1 model, with the addition of the retry parameter in the service rate, μ .

5.5 Developing latency models based on TH(O)RP

In the previous section a basic latency model was derived, along with more complex models. In this section the input parameters for the models are derived. The TH(O)RP routing protocol and various MAC protocols are analysed to define the appropriate input parameters.

5.5.1 Communications protocol breakdown

The current communication network of the CoCT is a circuit switched network. In the feasibility study of this thesis a packet switched network was proposed, because they are more robust and share network resources more effectively than traditional circuit switched networks.

The communication system for the network is based on the open systems interconnection (OSI) reference model, as seen in Figure 5.4 [45, 43]. Layers of interest are, the application layer, network layer, data link and physical layers.

The TRAFX protocol will reside in the application layer and the TH(O)RP protocol in the network layer. Any additional services included in the network will reside in the application layer. To get data through the network a routing protocol is required, which resides in the network layer. The communication technologies all have devices that use different protocols for the data link and physical layers. These are categorised under MAC protocols.

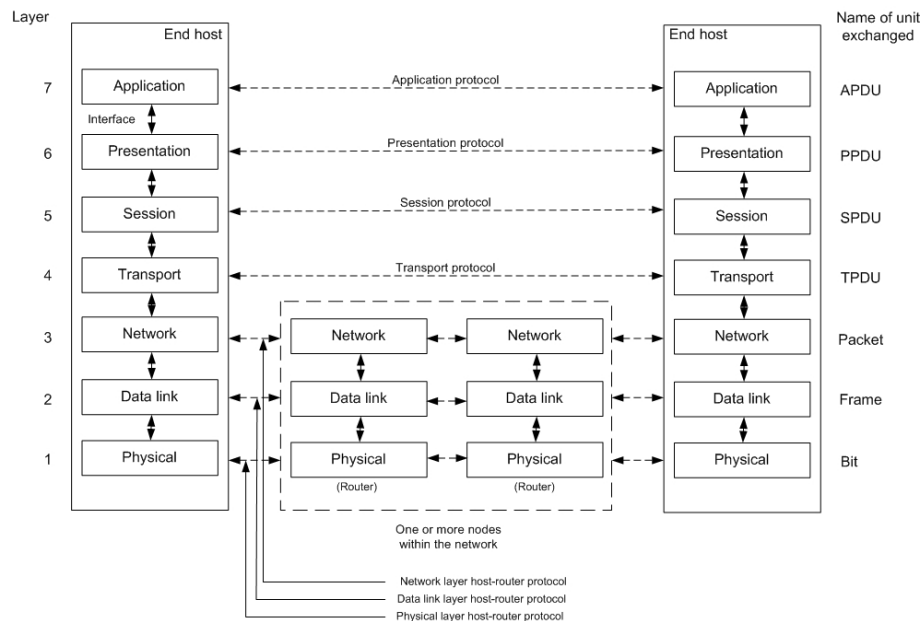


Figure 5.4: The OSI reference model [45]

5.5.2 Routing protocol

Brief overview

To manage the communications network and to route data traffic through the network, a routing protocol was developed in this project, namely the tree hierarchy (open) routing protocol (TH(O)RP). A detailed description of this can be found in Chapter 4.

TH(O)RP manages the network by constructing a tree hierarchy along which all traffic is routed. The characteristics of the protocol that will effect the latency models and how these characteristics can be incorporated into the models are briefly discussed here.

Control traffic and header overhead

In Table 5.1 the header sizes of the different TH(O)RP messages are summarised. The maximum payload sizes were calculated from an Ethernet payload of 1500 bytes as a reference. From this reference payload size, the associated headers of any messages used are subtracted.

Because all traffic generated by TH(O)RP is transmitted over UDP/IPv4 sockets, an initial combined UDP and IPv4 header is subtracted, which equals 28 bytes. Additionally, any TH(O)RP messages used introduces their associated headers which are also subtracted from the reference payload of 1500 bytes. See Figure 4.11 for the generic TH(O)RP packet, and Section 5.5.3 for an illustration of packet nesting. The control traffic header overhead is summarised in Table 5.1.

By following this method the maximum allowed size of a TRAFX payload can be derived as:

$$\begin{aligned}
 & Ethernet_{payload} - (UDP_{header} + IPv4_{header}) - TH(O)RP_{packet\ header} - \dots \\
 & \dots - TH(O)RP_{message\ header} - Data_{packet\ header} - TRAFX_{packet\ header} \\
 & = 1500 - 28 - 4 - 12 - 16 - 16 = 1424\text{ bytes}
 \end{aligned}$$

This defines the effective maximum payload size of a TRAFX message as 1424 bytes, but if the maximum allowed size is used, there will be no space left for other TH(O)RP messages to be included in a TH(O)RP packet which already contains a TRAFX message. The payload size of TRAFX messages were defined as approximately 800 bits in [25], and this payload size will be used in the latency models. See Appendix A for various packet and frame formats of protocols from the application layer through to physical layers.

Table 5.1: *TH(O)RP message and header sizes*

Category	MSG type	Size (payload) [bits]	Header [bits]
Control	TH(O)RP packet	Max. 11744	32
Control	TH(O)RP message	Max. 11648	96
Control	HELLO	384	32
Control	RREQ	128	32
Control	RREP	128	32
Control	MID	64	0
Control	HNA	64	0
Control/Data	ACK	0	64
Data	TRAFX	Max. 11520	128
Data	TRAFX	Real 800	128
Application	TRAFX	Max. 11392	128
Application	TRAFX	Real 672	128

Piggy backing

As can be seen from the TH(O)RP packet format, multiple messages can be contained in a single TH(O)RP packet, which allows piggy backing of multiple messages. This is an attempt to minimise the network traffic and leads to different combinations of messages that can reside in a TH(O)RP packet. For the purpose of calculating the maximum latency, a statistical indication of how much data can be generated per second is found, using the message generation rates of TH(O)RP. From this, the number of TH(O)RP packets required to transmit this data, and ultimately the amount of data that needs to be processed per second, are known. With this information a worst case scenario latency can be calculated. Actual latencies could be lower.

Forwarding jitter

Because communication mediums are usually shared, for instance a radio channel, a forwarding jitter has been introduced in the routing protocol to reduce collisions in the medium.

If it is assumed that the average jitter always applies, all generated messages will still arrive at their predefined generation intervals (See Figure 5.5(a)). Thus, the forwarding jitter has no effect on the message generation rates.

The forwarding jitter introduces additional latency when messages are being forwarded over multiple hops (See Figure 5.5(b)). It should be noted that the forwarding jitter is separate from the processing time and it applies once per hop for message transmission. Because of this, the forwarding jitter is added on top of the processing time.

The forwarding jitter used at a node, is a random value between 0 and the maximum jitter, and this value is re-calculated after each transmission interval. Because this is a random number, an even distribution between 0 and the maximum jitter can be assumed, therefore the average jitter is $\frac{jitter_{max}}{2}$. The latency incurred due to the forwarding jitter is:

$$t_{fwdjitter} = N \cdot \frac{jitter_{max}}{2} \tag{5.5.1}$$

where N is the number of hops along a communication route.

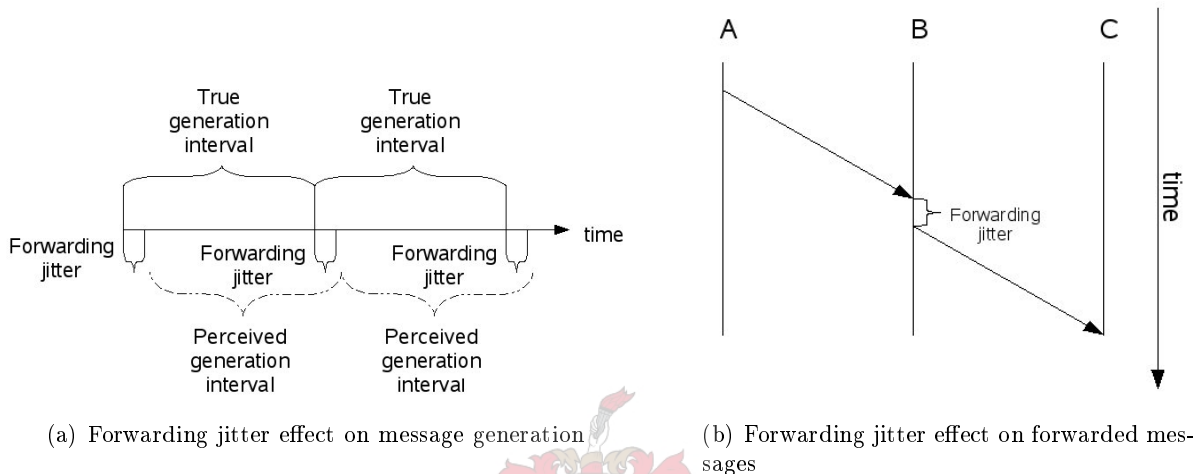


Figure 5.5: *The effects of the forwarding jitter*

Message generation rates

The control messages of the TH(O)RP protocol have static generation intervals, as does the TRAFX data packets. These rates are summarised in Table 5.2.

Table 5.2: *Generation rates*

Message	Amount	Rate [sec]	Number of arrivals/s
Hello	1	2	$\frac{1}{2}$
RREQ/RREP/ACK (ROUTE)	1	6	$\frac{1}{6}$
MID/HNA	1	5	$\frac{1}{5}$
Data/ACK (Data)	1	1	1

5.5.3 Packet nesting

When messages are transmitted over a network interface which has a protocol for each of the network layers, it results in nested messages. This adds to the overhead of transmitting messages over a network. See Figure 5.6 for an example where a TRAFX payload is encapsulated in a TRAFX compatibility packet. This in turn is inside a TH(O)RP data message, which is inside a TH(O)RP packet. The message is then encapsulated in a UDP packet, which in turn is inside an IPv4 datagram. When this is passed to the network interface, in this case an 802.11 interface, it is encapsulated in an 802.11 frame.

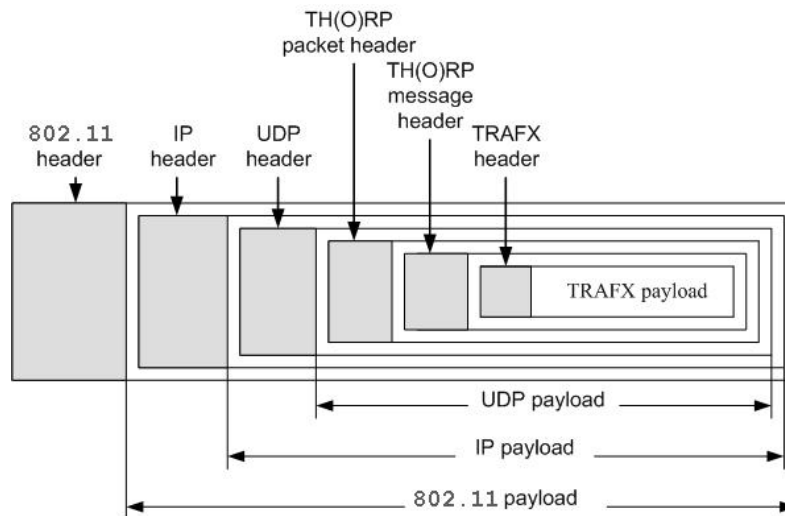


Figure 5.6: Packet nesting

5.5.4 Transport- and MAC protocol overhead

As described in the previous section, messages are encapsulated in other messages as they are passed along the different network layers. In Table 5.3², the different communication technologies are summarised, along with their respective header and data payload sizes. In order to simplify the latency models used, all protocols are assumed to have the same payload sizes as the Ethernet MAC, namely 1500 bytes. The control traffic and wait times of the MAC protocols are also not included.

Table 5.3: Transport- and MAC protocol header sizes

Protocol	Header [bits]	Optional [bits] Assume $n = 0$	Maximum transmission unit (MTU) [bits]	MTU [bytes]
UDP	64	N/A	MAC dependant	MAC dependant
TCP	160	$n*32$	MAC dependant	MAC dependant
IP	160	$n*32$	MAC dependant	MAC dependant
OC		N/A		
Ethernet	72	N/A	12000	1500
802.11	272	N/A	18496	2312
802.16	51	N/A		
GPRS (single slot)	42	N/A	114	14
GPRS (all slots)	8400	N/A	22800	2850

5.6 Latency models

To calculate the latency of the TH(O)RP routing protocol and respective MAC protocols, we substitute the following values into the latency model parameters.

Define λ as the generation- or arrival rate of all messages at a per second rate. In Table 5.4, λ is calculated by adding the generation rates for the HELLO, MID and HNA, ROUTE and

²In Table 5.3 n denotes the number of optional 32 bit words in TCP and IP headers.

ACK (ROUTE), and DATA and ACK (DATA) messages together, for instance:

$$\lambda = \frac{1}{2} + 2 \cdot \frac{1}{5} + 2 \cdot \frac{1}{6} + 2 = 3\frac{7}{30}$$

The service rate is defined as $\mu = \frac{BW}{D_t}$, where D_t is the total amount of generated data in bits/s, and BW is the device bandwidth in bits/s. The stability of the model is calculated with ρ and the processing wait time is calculated as described before. These calculate the processing- and transmission times of the generated data. The forwarding jitter is added as a separate entity on top of the processing- and transmission time. The total latency is given by:

$$T_{wait_jitter} = T_{wait} + t_{fwdjitter} \quad (5.6.1)$$

The resulting parameters are summarised in Tables 5.4 and 5.5³. These model parameters can be used with the different device bandwidths to calculate the latency for each technology. Although the TH(O)RP routing protocol was primarily developed to be used over a multi-hop wireless network, it can also be used on wired and cellular networks. There are, however, more efficient protocols for use on wired and cellular networks, but it is beyond the scope of this thesis to calculate the latencies for those protocols. TH(O)RP can be optimised for different networks. For instance, in a static wired network there is little possibility of links being broken, so control traffic of the routing protocol can be generated at much lower rates, which will reduce the amount of control traffic in the network. This is, however, not discussed here, and the latency results will be compiled from parameters optimised for use in a wireless environment.

Because the TRAFX messages are encapsulated inside a TH(O)RP data message, it is not separately included in the latency models as it is indirectly included through the data messages wherein it is encapsulated.

To calculate latencies for different bandwidths and models, a Matlab application, the code of which is shown in Appendix D, was written to compute the latencies, based on the input parameters supplied for the application. As input, a device bandwidth, forwarding jitter, channel error rate, model type and network interface type are provided and the application will calculate the resulting latency. This application was directly based on the latency model parameters in Tables 5.4 and 5.5.

³The amount of carry over data is determined by the number of hops. Thus for N-hops, N-1 carry over data components are added. When the carry over data comes from multiple sources (N:N:1 upward scenario), it is multiplied by the number of sources.

Table 5.4: Latency model parameters for TH(O)RP routing protocol ($N=2$)

Model & Direction	λ	μ	ρ	T_L	Data (D)
1:1 Up	$3\frac{7}{30}$	$\frac{BW}{D}$	$\frac{\lambda}{\mu}$	$\frac{1}{\mu-\lambda}$	$D_{Up} + D_H$
1:1 Down	$3\frac{7}{30}$	$\frac{BW}{D}$	$\frac{\lambda}{\mu}$	$\frac{1}{\mu-\lambda}$	$D_{Down} + D_H$
N:1 Up	$\sum_{n=1}^N \lambda_n = 6\frac{7}{15}$	$\frac{BW}{D}$	$\frac{\lambda_1+\lambda_2}{\mu}$	$\frac{1}{\mu-(\lambda_1+\lambda_2)}$	$D_{Up} + D_H$
1:N Down	$\lambda_{1,2} = 3\frac{7}{30}$	$\mu_{1,2} = \frac{BW}{D}$	$\frac{\lambda_{1,2}}{\mu_{1,2}}$	$\frac{1}{\mu_{1,2}-\lambda_{1,2}}$	$D_{Down} + D_H$
1:1:1 Up	$\lambda_{1,2} = 3\frac{7}{30}$	$\mu_1 = \frac{BW}{D_1}$ $\mu_2 = \frac{BW}{D_2}$	$\frac{\lambda}{\mu_1}$ $\frac{\lambda}{\mu_2}$	$\sum_{n=1}^N T_{L_n}$ $T_{L_{1,2}} = \frac{1}{\mu_{1,2}-\lambda}$	$D_1 = D_{Up} + D_H$ $D_2 = D_{Up} + D_H + D_{CO}$
1:1:1 Down	$\lambda_{1,2} = 3\frac{7}{30}$	$\mu_1 = \frac{BW}{D_1}$ $\mu_2 = \frac{BW}{D_2}$	$\frac{\lambda}{\mu_1}$ $\frac{\lambda}{\mu_2}$	$\sum_{n=1}^N T_{L_n}$ $T_{L_{1,2}} = \frac{1}{\mu_{1,2}-\lambda}$	$D_1 = D_{Down} + D_H$ $D_2 = D_{Down} + D_H + D_{CO}$
N:N:1 Up	$\lambda_I = \sum_{n=1}^N \lambda_n = 6\frac{7}{15}$ $\lambda_{II} = \lambda_I$ $\lambda_{III} = \lambda_I + \lambda_{II}$ $= 12\frac{14}{15}$	$\mu_1 = \frac{BW}{D_1}$ $\mu_2 = \frac{BW}{D_2}$ $\mu_3 = \frac{BW}{D_3}$	$\rho_1 = \frac{\lambda_I}{\mu_1}$ $\rho_2 = \frac{\lambda_{II}}{\mu_2}$ $\rho_3 = \frac{\lambda_{III}}{\mu_3}$	$T_{L_{I,II}} + T_{L_{III}}$ $T_{L_{I,II}} = \frac{1}{\mu_{1,2}-\lambda_{I,II}}$ $T_{L_{III}} = \frac{1}{\mu_3-\lambda_{III}}$	$D_1 = D_{Up} + D_H$ $D_2 = D_1$ $D_3 = D_1 + D_2 + 2 \cdot D_{CO}$
1:N:N Down	$\lambda_{1,2,3,\dots} = 3\frac{7}{30}$	$\mu_1 = \frac{BW}{D_1}$ $\mu_2 = \frac{BW}{D_2}$ $\mu_3 = \frac{BW}{D_{3,4,5,6}}$	$\frac{\lambda}{\mu_1}$ $\frac{\lambda}{\mu_2}$ $\frac{\lambda}{\mu_3}$	$T_{L_{I,II}} + T_{L_{III}}$ $T_{L_{I,II}} = \frac{1}{\mu_{1,2}-\lambda}$ $T_{L_{III}} = \frac{1}{\mu_3-\lambda}$	$D_1 = D_{Down} + D_H$ $D_2 = D_1$ $D_{3,\dots} = D_{Down} + D_H + D_{CO}$
Retry Up	λ	$\frac{BW}{D}$	$\frac{\lambda}{\mu}$	$\frac{1}{\mu-\lambda}$	$(1 + p_{err}) \cdot (D_{Up} + D_H)$
Retry Down	λ	$\frac{BW}{D}$	$\frac{\lambda}{\mu}$	$\frac{1}{\mu-\lambda}$	$(1 + p_{err}) \cdot (D_{Down} + D_H)$

Table 5.5: Resulting data sizes

Direction	Data	Breakdown of data sources	Total Bits
Single hop up	D_{Up}	$\frac{1}{2}(384) + \frac{4}{10}(64) + \frac{2}{12}(128) + (800)$	1038.9333
	D_H	$\frac{1}{2}(32 + 96) + \frac{4}{10}(96) + \frac{2}{12}(32 + 64 + 2 \cdot 96) + (128 + 64 + 2 \cdot 96)$	534.4
Total ($D_{Up} + D_H$)		Add TH(O)RP- (32), IPv4_UDP- (224) and 802.11 headers (272)	2101.3333
Single hop up	D_{Down}	$\frac{1}{2}(384) + \frac{4}{10}(64) + \frac{2}{12}(128) + (800)$	1038.9333
	D_H	$\frac{1}{2}(32 + 96) + \frac{4}{10}(96) + \frac{2}{12}(32 + 64 + 2 \cdot 96) + (128 + 64 + 2 \cdot 96)$	534.4
Total ($D_{Down} + D_H$)		Add TH(O)RP- (32), IPv4_UDP- (224) and 802.11 headers (272)	2101.3333
Multi-hop up	D_{Up}	$\frac{1}{2}(384) + \frac{4}{10}(64) + \frac{2}{12}(128) + (800)$	1038.9333
	D_H	$\frac{1}{2}(32 + 96) + \frac{4}{10}(96) + \frac{2}{12}(32 + 64 + 2 \cdot 96) + (128 + 64 + 2 \cdot 96)$	534.4
	D_{CO}	$\frac{4}{10}(64 + 96) + \frac{2}{12}(128 + 32 + 64 + 2 \cdot 96) + (800 + 128 + 64 + 2 \cdot 96)$	1317.333
<i>1:1:1 & N:N:1</i>		Add TH(O)RP- (32), IPv4_UDP- (224) and 802.11 headers (272)	3418.6663
Total ($D_{Up} + D_H + D_{CO}$)			
Multi-hop down	D_{Down}	$\frac{1}{2}(384) + \frac{4}{10}(64) + \frac{2}{12}(128) + (800)$	1038.9333
	D_H	$\frac{1}{2}(32 + 96) + \frac{4}{10}(96) + \frac{2}{12}(32 + 64 + 2 \cdot 96) + (128 + 64 + 2 \cdot 96)$	534.4
	D_{CO}	$\frac{4}{10}(64 + 96) + \frac{2}{12}(128 + 32 + 64 + 2 \cdot 96) + (8320 + 128 + 64 + 2 \cdot 96)$	1317.333
<i>1:1:1 & 1:N:N</i>		Add TH(O)RP- (32), IPv4_UDP- (224) and 802.11 headers (272)	3418.6663
Total ($D_{Down} + D_H + D_{CO}$)			

Chapter 6

TH(O)RP simulation setup

6.1 Introduction

6.1.1 Simulation overview

The theoretical analysis and prediction of the behaviour of network protocols operating over large number of nodes, and over multiple hops have always been a challenging process. This is in part due to the manner in which these protocols need to cooperate via message passing. Some messages get lost, others arrive out of order, some might even arrive at the wrong recipient.

This prompts us to simulate these network protocols in order to verify their correct operation under as many circumstances as possible. Through simulation the efficiency of these protocols can be measured, and they can be optimised and improved. By graphically representing a protocol's operation, it can be visually verified, and it can be seen when and why the protocol fails. This points out the merits and shortcomings of a protocol, and allows problems to be fixed, or alternative solution to be suggested.

The chosen simulation environment for this project is the OMNET++ discrete event simulation environment and the mobility framework, as discussed in the following section.

6.1.2 OMNET++ and the mobility framework

To simulate the operation of the TH(O)RP routing protocol, a network simulator was required. The OMNET++ discrete event simulation system [48] was chosen, which is an extensible simulation environment written in C++.

One of the extensions to OMNET++ is the mobility framework [38]. It provides a framework for simulating a wireless network, using an 802.11 MAC layer, where connections between nodes are set up dynamically according to whether or not they are within range of each other. With the mobility framework the efficiency of TH(O)RP could be measured in a wireless environment, in keeping with the proposed solution for this project.

OMNET++ allows the graphical representation of a network with multiple nodes, displaying network packet transmissions and node placement in the network. It also allows for real time monitoring of collected simulation statistics.

The mobility framework allows the definition of an 802.11 MAC and physical layer, and separate network and application layers with methods to accomplish message passing between the layers. It also provides a framework for defining TH(O)RP network messages, and func-

tions to handle these messages. The software versions used are OMNET++3.2p1 and mobility framework1-0a6. See Figure 6.1 for an example simulation setup, displaying node placement and a single 802.11 airframe that is being transmitted.

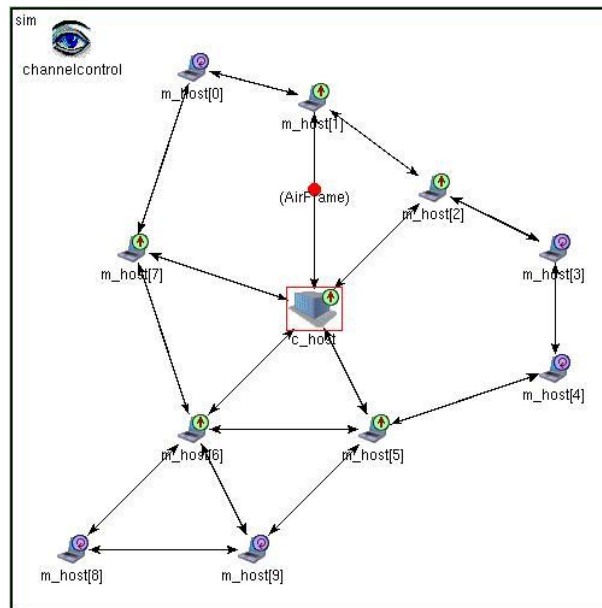


Figure 6.1: Example simulation setup with 10 outstations and central instation

6.1.3 Modus Operandi

First, the TH(O)RP protocol was implemented in C++, and made to be compatible with the OMNET++ and mobility framework environments.

After TH(O)RP was implemented in the simulation environment, some initial simulations were executed to verify its correct operation. After the necessary changes were made, further tests were executed to optimise the parameters of the routing protocol. Finally, additional functionality was added to improve the operation, specifically responsiveness to network changes, of the protocol.

6.1.4 Summary

The goal of this chapter is to set up the simulation of TH(O)RP, in order to measure its efficiency and to develop improvements. The simulated implementation of the protocol will also provide a good platform for future developments and add-ons to the protocol.

6.2 Model overview and simulation environment

6.2.1 Overview

The simulation can be divided into two separate entities. The first and most general is the simulation environment, or playground, wherein the nodes are situated. The second is the nodes of the network, which can further be broken up into a physical, network and application layer.

The simulation environment is used to define different scenarios and control the communications medium over which messages are transmitted.

Due to the nature of the proposed system, it will not be mobile, thus nodes will stay in their initial positions for the duration of the simulation. Nodes can however fail or shutdown any time during the simulation. This is because of interruptions that might occur in the real-life implementation of the proposed solution. For instance, a node could be shut down for maintenance, or a physical obstruction like a bus could disrupt two communicating nodes. Due to nodes failing, some routes may need to be repaired and different routes constructed. This gives the network a dynamic characteristic, where nodes are not mobile but might need to reconstruct routes from time to time.

6.2.2 Node composition

There are two types of network nodes defined, a TIC (outstation) and the central controller (instation). There will always be a single instation and one or many outstations. The instation will be at the top of the network hierarchy, where the outstations will form around the instation to create the tree hierarchy of the TH(O)RP protocol. The instation will remain at the top of the hierarchy throughout the entire simulation.

Each node is broken up into three different layers, a physical layer, network layer and application layer. The physical layer corresponds to an 802.11 MAC. The network layer is an implementation of the TH(O)RP protocol. The application layer is a functionally limited representation of the TRAFX protocol [25] used by the CoCT. In Figure 6.2, the graphical representation of the different layers of the nodes can be seen.

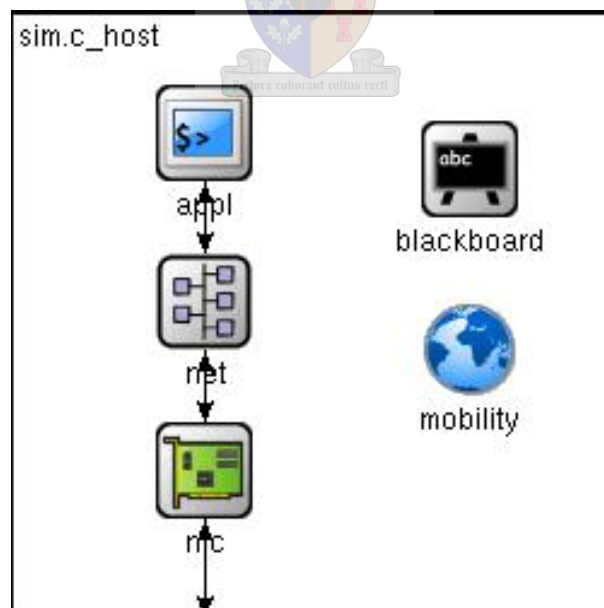


Figure 6.2: The different layers of the network nodes

6.2.3 Layers of the nodes

The nodes are composed of three layers, as seen in Figure 6.2. The bottom layer, network interface card (NIC), is an implementation of the 802.11 wireless network interface. This layer provides the physical layer functionality and handles radio collisions, back-offs and radio states. It also encapsulates the network layer traffic into physical layer messages that can be transmitted over the air in a frame format.

The network layer is an implementation of the TH(O)RP routing protocol. It sets up the tree hierarchy used by TH(O)RP, uses forwarding jitter to minimise collisions in the communications medium, and incorporates a best effort transport mechanism with multiple message retransmissions. It also incorporates an acknowledgement system, used to facilitate successful message delivery over multiple hops in a lossy environment. The network layer also informs the application layer of nodes that are reachable in the network and ensures that a message is only passed to the application layer of a node to whom a message is addressed. See Figure 6.3 for the network layer output of the instation¹ and one outstation in the example network of Figure 6.1. The network layer prints the contents of the different information repositories kept by TH(O)RP nodes.

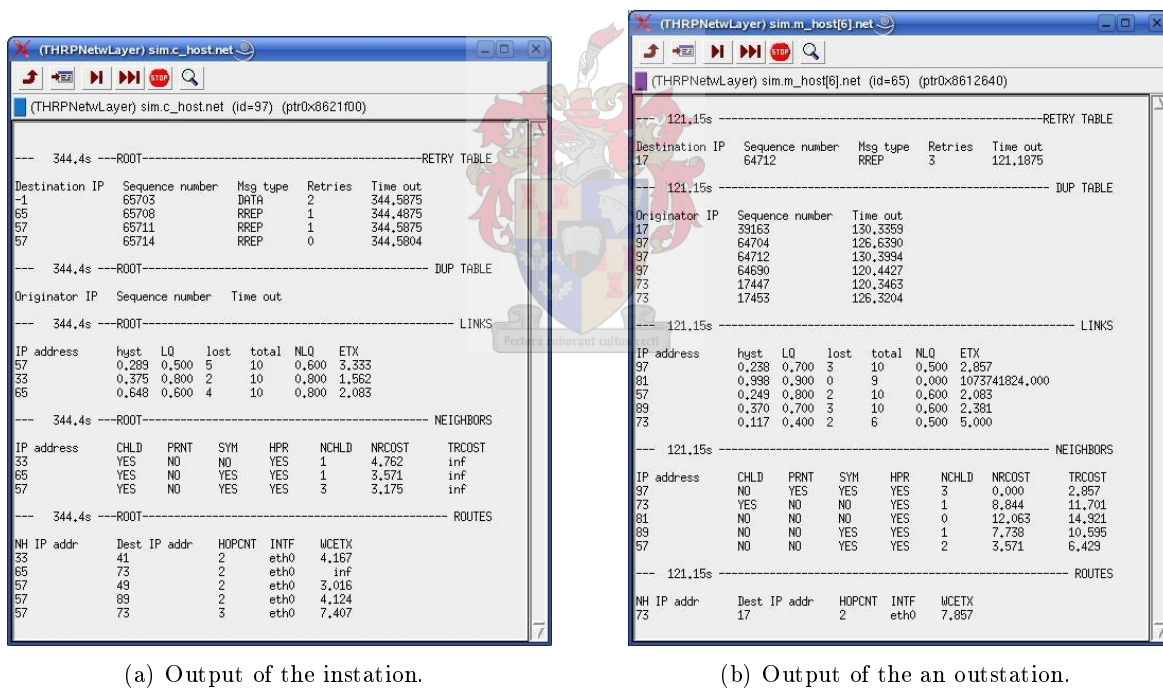


Figure 6.3: Network layer output during simulation

The application layer is a functionally limited implementation of the TRAFX protocol [25] used by the CoCT to communicate with their TICs. It is functionally limited because it only generates empty TRAFX messages, with a bogus payload of approximately the same size than real TRAFX messages. The TRAFX implementation also differs for the two different types of nodes in the network. The instation is the only node that can generate TRAFX requests, and the outstations will only respond to a request if it received a request addressed to it. The goal of such a TRAFX implementation was to measure the response time of TRAFX messages over

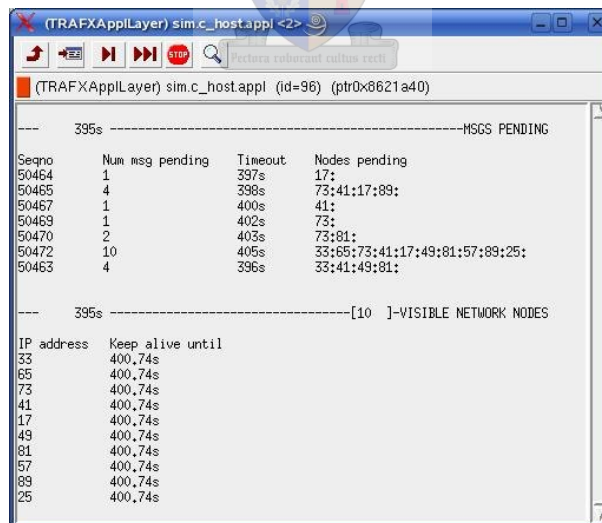
¹Note the ROOT identifier in the output of the instation.

the TH(O)RP protocol over a multi-hop network. See Figure 6.4 for application layer output of the instantiation². The information represented in this output gives us an indication of the number of nodes that are visible to the instantiation, along with their network addresses. It can also be seen how many TRAFX messages are currently pending, thus how many TRAFX messages still need to be replied by the outstations of the network, and which nodes still need to reply to TRAFX requests. A timer indicating when a pending message will be considered lost, has been included as well.

The implementation of the TRAFX protocol allows us to generate either dialogue or polylogue messages. A dialogue message is a broadcast message that is intended for all the nodes in the network. When an outstation receives a dialogue message, it will generate a TRAFX reply and forward the dialogue message to all its children, if any. If a node receives a polylogue message it will either send a TRAFX reply if it is the intended recipient, or it will forward the message to the intended destination if it has a route to that destination, else it will simply drop the message.

The TRAFX implementation randomly decides to generate either a dialogue or polylogue message. It also randomly chooses recipients for these messages, in the case of a polylogue message, from a table of visible nodes that it maintains. This table is updated from the TH(O)RP network layer every time the instantiation receives a route request message.

Under normal conditions, the TRAFX implementation will generate both dialogue and polylogue messages and this default will be used in the initial simulations. For more specific tests the TRAFX protocol can be informed to only generate polylogue messages, and to which node these messages should be addressed. This allows the success rate and latency for a specific node, and a specific number of network hops to be measured. This was the case with the multi-hop and protocol responsiveness tests.



```

--- 395s -----MSGS PENDING
Seqno  Num msg pending  Timeout  Nodes pending
50464  1                 397s    17:
50465  4                 398s    73:41:17:89:
50467  1                 400s    41:
50469  1                 402s    73:
50470  2                 403s    73:81:
50472  10                405s    33:65:73:41:17:49:81:57:89:25:
50463  4                 396s    33:41:49:81:

--- 395s -----[10 ]-VISIBLE NETWORK NODES
IP address  Keep alive until
33          400,74s
65          400,74s
73          400,74s
41          400,74s
17          400,74s
49          400,74s
81          400,74s
57          400,74s
89          400,74s
25          400,74s

```

Figure 6.4: Application layer output of instantiation.

²The outstations do not have any application layer output.

6.2.4 Network messages

The TH(O)RP messages were easily defined in the OMNET++ environment due to the message template offered (See Appendix B). Network messages are defined in a ‘.msg file’, and when the simulation is compiled, the relevant C++ source and header files are created with getter and setter methods for the various fields of the different messages. Regardless of this help from the OMNET++ environment, network packets that could contain multiple TH(O)RP messages were required. This was required to enable piggybacking of multiple messages in a single network packet, used to reduce the amount of control traffic in the network.

After some initial trials, the definition of abstract fields in the base class of the main network packet was achieved. This allowed the modification of the generated message definitions, to incorporate a list of TH(O)RP messages that would be included in a single network packet. See Figure 6.5(a) for a network packet containing multiple TH(O)RP messages, specifically a data message, a hello message with neighbour information and a route reply message. Fortunately, OMNET++ provides an excellent method for encapsulating messages inside each other, which allowed the encapsulation of the application layer TRAFX messages in the network layer TH(O)RP messages. A TRAFX request message can be seen in Figure 6.5(b).

6.2.5 Node status colour coding and bubble messages

To simplify the analysis of the TH(O)RP protocol, it was decided to add coloured states to the nodes. There are three states that a node can be in, shutdown, active but orphaned and active with a parent. The colour codes for these states are red, blue and green respectively. Introducing coloured states, allowed for easy identification of shutdown nodes, nodes that are active but not part of the tree hierarchy and nodes that are part of the tree hierarchy. The instation will always start with a green state because it is the root of the tree hierarchy. All other nodes start in the blue state, or if it was turned off at the beginning of the simulation it will start in the red state.

A further visual aid was introduced by the implementation of helpful information being displayed above each node during certain events, specifically message- generation and forwarding events. These information bubbles can be switched on or off. There are different message groups for the information bubbles, namely: DATA messages, ROUTE messages (RREQ, RREP, RERR), MID messages, HNA messages, ACK messages and HELLO messages. Using these information bubbles, the protocol can be visually verified. For instance, it can be verified whether data messages are routed along an intended route towards a desired destination.

Both of these visual aids are only available in the graphical simulation environment of OMNET++, namely TkEnv. See Figure 6.6 for examples of the visual aids.

6.2.6 Graphical tree hierarchy representation

The most illustrative method to view the tree hierarchy constructed by the nodes of the TH(O)RP protocol is to display it graphically. This proved to be a real challenge, due to the fact that no single node in the network has a full view of what the network topology looks like. The only information available was the local information that each node collects about its neighbourhood, and the entries in its routing table. This information needed to be collected and merged into a single entity to form a global view of the network topology.

```

(ThrpPkt) ...T.Generic THRP packet (encapsulated msg)
(ThrpPkt) simulation.scheduled-events..BROADCAST.Generic THRP packet
General | Sending/Arrival | Fields | Control Info | Params |
class ThrpPkt {
  int destAddr = -1
  int srcAddr = 97
  int ttl = 1888
  unsigned long seqNum = 2868
  ThrpMsg msg[0] = {
    int msgType = 8
    double vTime = 0.000000
    int msgSize = 28
    int originator = 97
    int ttl = 255
    int hopCnt = 0
    int msgSeqNum = 65243
    dataMsg data[0] = {
      int msgType = 0
      char flags = 0
      int msgSize = 0
      int next_hop = 65
      int src = 97
      int dest = 65
    }
  }
  ThrpMsg msg[1] = {
    int msgType = 1
    double vTime = 6.000000
    int msgSize = 44
    int originator = 97
    int ttl = 1
    int hopCnt = 0
    int msgSeqNum = 65244
    lqHelloMsg lqHello[0] = {
      double htime = 2.000000
      int will = 3
      int haveparent = 1
      int numchild = 2
      double rootcost = 0.000000
      int paddr = 97
      neighInfo neigh[0] = {
        int linkType = 3
        int neighType = 0
        double linkQuality = 0.600000
        double neighLinkQuality = 0.000000
        int addr = 57
      }
      neighInfo neigh[1] = {
        int linkType = 1
        int neighType = 0
        double linkQuality = 0.700000
        double neighLinkQuality = 0.000000
        int addr = 33
      }
      neighInfo neigh[2] = {
        int linkType = 3
        int neighType = 0
        double linkQuality = 0.700000
        double neighLinkQuality = 0.500000
        int addr = 65
      }
    }
  }
  ThrpMsg msg[2] = {
    int msgType = 5
    double vTime = 30.000000
    int msgSize = 40
    int originator = 97
    int ttl = 255
    int hopCnt = 0
    int msgSeqNum = 65245
    rrepMsg rrep[0] = {
      char flags = 0
      int rrepSeqNum = 25714
      int htime = 6
      int status = 0
      double ucctx = 0.000000
      int next_hop = 57
      int src = 97
      int dest = 57
    }
  }
}
}

```

(a) TH(O)RP packet with multiple message

```

(TrafXpkt) ...ket (encapsulated msg).Generic TRAFX msg <
(ThrpPkt) simulation.scheduled-events..BROADCAST.Generic THRP pa
General | Sending/Arrival | Fields | Control Info | Params |
class TrafXpkt {
  int destAddr = 65
  int srcAddr = 97
  int msgSeqNum = 50379
  int msgType = 0
  int msgSize = 0
  double msgTimeStamp = 302.000000
  payload body = {
    char content[0] = 35
    char content[1] = 35
    char content[2] = 35
    char content[3] = 35
    char content[4] = 35
    char content[5] = 35
    char content[6] = 35
    char content[7] = 35
    char content[8] = 35
    char content[9] = 35
    char content[10] = 35
    char content[11] = 35
    char content[12] = 35
    char content[13] = 35
    char content[14] = 35
    char content[15] = 35
    char content[16] = 35
    char content[17] = 35
    char content[18] = 35
    char content[19] = 35
    char content[20] = 35
    char content[21] = 35
    char content[22] = 35
    char content[23] = 35
    char content[24] = 35
    char content[25] = 35
    char content[26] = 35
    char content[27] = 35
    char content[28] = 35
    char content[29] = 35
    char content[30] = 35
    char content[31] = 35
    char content[32] = 35
    char content[33] = 35
  }
}

```

(b) Encapsulated TRAFX packet

Figure 6.5: Illustration of TH(O)RP packet and contents

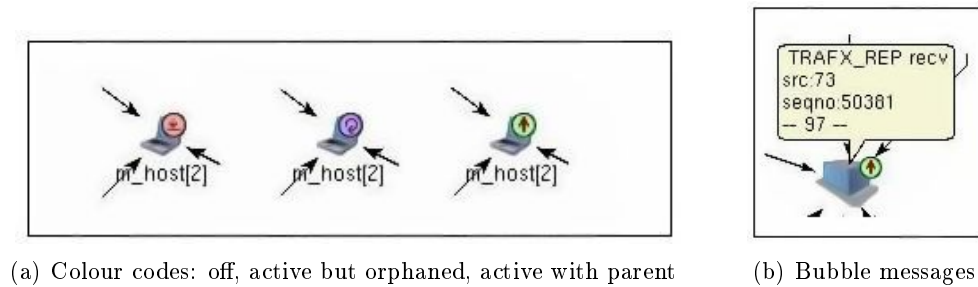


Figure 6.6: *Simulation visual aids*

Another challenge was how to represent the information in the tables of the nodes graphically. This was accomplished with the graphviz package and the dot application from AT&T Research and Lucent Bell Labs [33]. The dot package provides a language for defining directed graphs in text form, and then it translates the text into a graphical representation.

To accomplish the two challenges mentioned above, a script called ‘create_dot_graph’ was written, that could collect information stored in ‘.dot’ files generated by each node when a simulation ends. In the ‘.dot’ files a node records information about its parent, its neighbours and the ETX metric to each of its neighbours. Using this information, a global understanding of the network topology could be generated, and it could be verified that the TH(O)RP protocol chooses optimal routes to the instation.

The ‘create_dot_graph’ script generates output graphs in the POST-SCRIPT, JPEG and PDF formats and leaves the resulting ‘.dot’ file behind, which can be modified to improve the generated graphs. By default, the ‘create_dot_graph’ script only generates output graphs with connections to the parent nodes. This is due to the fact that graphs which show all node connections tend to be chaotic and it becomes difficult to derive any structure from them. The graphs with connections to only the parents forms a well behaved tree hierarchy, as expected.

The ‘create_dot_graph’ script has an input parameter, ‘-a’, which can force it to also create output graphs that contains links to all neighbours. This is useful to verify that the tree hierarchy is built around optimal routes between the outstations and the instation. This option should be used with caution because in networks with many nodes the dot package might not be able to generate the desired output graphs and could stall indefinitely. This is due to a large amount of nodes and edges that create complex graphs, and laying them out becomes too complex to solve. See Figure 6.7 for the output of the ‘create_dot_graph’ script.

Using this script a graphical representation of the network topology can be automatically generated, for any simulation, within certain constraints. The script could not generate valid graphs for networks with more than approximately 100 nodes. The resulting graphs can also be misleading at times, as they represent what was recorded in the tables of the nodes at the precise time that the simulation ended. This can cause the resulting graph to indicate that a shorter route exists than the current route used by a node. This is due to the periodic nature of the TH(O)RP protocol and the fact that the simulation ended before it could update its routes.

Regardless of these limitations, it does generate impressive graphs. In Figure 6.8, the resulting output graphs for the example simulation shown in Figure 6.1, can be seen. In the graph, the instation is represented by the square node at the top of the graph, the ellipse nodes are the outstations, and the ETX metric is indicated on the edges between the nodes.


```

create_dot_graph v1.0
Overview: Creates a tree hierarchy graph from seperate dot files
Usage:    Normally only generates output graph with link to only parent,
          Run with '-a' option to generate links to all neighbors (for smaller networks)
Packages: Output generated by the 'Graphviz - Graph Drawing Programs
          from AT&T Research and Lucent Bell Labs' (dot) package
Packages: PDF output generated by eps2pdf of Wouter Kager
Packages: The 'grep' and 'cat' packages of the Free Software Foundation
Author:   Niel Morrison
Date:    15/10/2006

Merging dot files
#####

Creating dot output graphs (jpg, eps and pdf formats)

--- Creating jpeg outputs ---
##
--- Creating post script outputs ---
##

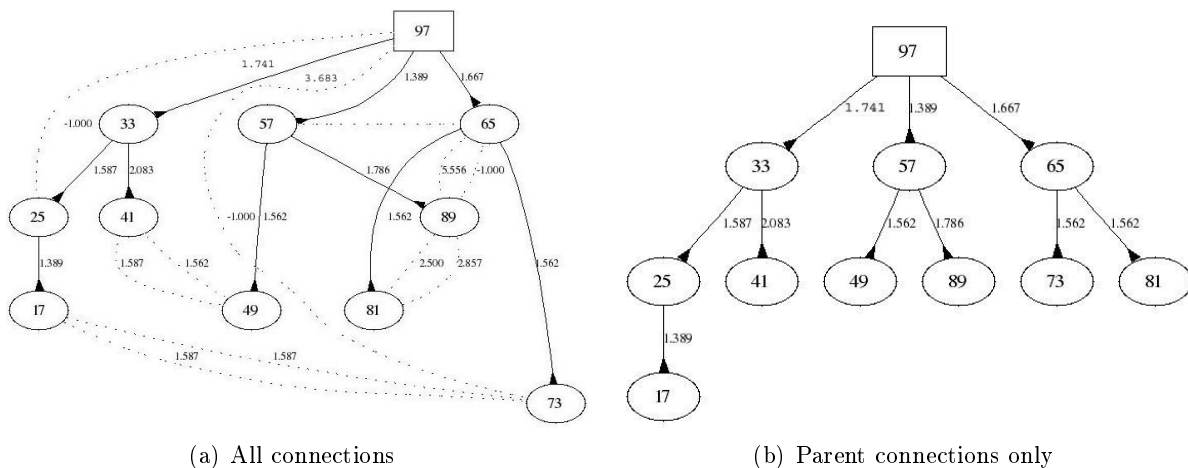
--- Creating pdf outputs ---
eps2pdf 2006/06/01 by Wouter Kager

eps2pdf: processing file /home/morrisondu/Development/workspace/simulation/mobile_thrp/d
eps2pdf: /home/morrisondu/Development/workspace/simulation/mobile_thrp/dot/outputDotGrap
nt/workspace/simulation/mobile_thrp/dot/outputDotGraph.pdf [ok]
-----

Notes: 1) PDF best viewed in KGhostView
       2) The dot output file can be modified to cleanup
          the generated graphs (see dot documentation).
       3) If the script was executed with the '-a' option
          then there are two output graphs, one with only links to
          parents included, and another with all links included.

Dot graphs generated successfully, view and enjoy!!

```

Figure 6.7: Text output of `create_dot_graph` script.Figure 6.8: Graphical output graphs created by `'create_dot_graph'` script

6.2.7 Simulation statistics and results collection

In order to measure the efficiency of the TH(O)RP protocol, statistical information about a simulation need to be gathered.

The information gathered relates to the number of TH(O)RP packets generated and the individual TH(O)RP messages generated, which are grouped into two categories, namely control and data traffic. From this, the average number of TH(O)RP messages inside a TH(O)RP packet can be calculated.

The number of bits associated with control traffic and the data traffic are calculated. The control bandwidth, data bandwidth and remaining bandwidth is measured. The ratio between data traffic and control traffic is also calculated.

The number of dropped packets, the drop rate per second, and the average number of re-transmissions required to successfully transmit a message, is calculated.

At the instation, the number of TRAFX requests generated, the number of TRAFX replies received and the number of requests not acknowledged, which indicates a TRAFX message loss, are calculated. Also, the number of data packets assumed to still be in the network, are measured. From these values, the average success rate or throughput of the network, is calculated. See Figures 6.9 and 6.10 for the output statistics after a 20 minute simulation run for the instation and two outstations of the example simulation from Figure 6.1.

```

Session Edit View Bookmarks Settings Help
-----
Application layer statistics [971]
Total sent packets: 4815
Total recv packets: 4665
Total lost packets: 148
Total packets in network: 2
Transmission success rate: 96.88474%
Pending msg percentage: 0.04154%
-----
Network layer statistics [971]
Thrp packets generated: 6788
Control Thrp messages generated: 22525
Data Thrp messages generated: 2141
Average number of messages per packet 3.6338

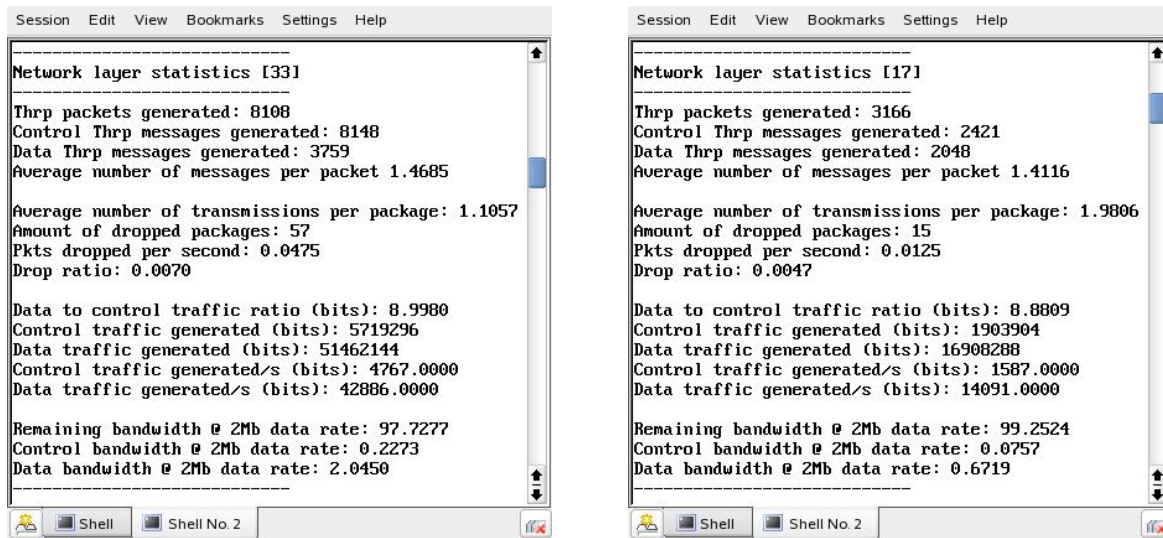
Average number of transmissions per package: 1.8683
Amount of dropped packages: 114
Pkts dropped per second: 0.0950
Drop ratio: 0.0168

Data to control traffic ratio (bits): 2.3628
Control traffic generated (bits): 7423040
Data traffic generated (bits): 17539072
Control traffic generated/s (bits): 6186.0000
Data traffic generated/s (bits): 14616.0000

Remaining bandwidth @ 2Mb data rate: 99.0081
Control bandwidth @ 2Mb data rate: 0.2950
Data bandwidth @ 2Mb data rate: 0.6969
-----
Shell Shell No. 2

```

Figure 6.9: Collected statistics for example simulation for the instation



(a) Outstation one

(b) Outstation two

Figure 6.10: *Collected statistics for example simulation for outstations*

To measure the response or latency of the network, the latency incurred between generating a TRAFX request and receiving a TRAFX reply, is measured. This allows the average latency of the network to be calculated. The latencies are stored in a vector output file, namely ‘omnetpp.vec’, which records the latency of each TRAFX message successfully received. The resulting latencies are plotted using the ‘plove’ [48] tool provided by OMNET++. During tests of the response to network disturbances, the dead period between the reception of TRAFX replies, is used to measure the approximate recovery time of the network hierarchy and routes.

6.2.8 Route table storage requirements

To measure the storage requirements of a node, the table with the greatest potential to get relatively large, namely the routing table, was analysed. This storage requirement is calculated from the routing table definition as specified in the C implementation of TH(O)RP.

Upon initialisation of the routing table, it occupies 32 address locations, which are each 32 bits. Thus, the empty routing table occupies:

$$32 \cdot 32 = 1024 \text{ bits} \quad (6.2.1)$$

Assuming that M destinations, can be reached through N routes, where $N \leq M$, the storage requirement for all routes of a node is:

$$N \cdot (152 + M \cdot 168) \quad (6.2.2)$$

The total route table storage requirement is:

$$RT_{storage} = 1024 + N \cdot (152 + M \cdot 168) \quad (6.2.3)$$

This equation, is used to calculate the approximate route storage requirements of a node, to test the scalability of TH(O)RP.

6.2.9 Simulation setup

The simulation setup is configured by three input files, namely ‘omnetpp.ini’, ‘parameters.ini’ and ‘layout.ini’. The general input file, ‘omnetpp.ini’, defines general settings for the simulation and also provides default settings for cases where the ‘parameters.ini’ and layout.ini files are omitted. The configuration files can be used to define the parameters of a simulation, as well as output filenames, simulation end-time and more. The ‘omnetpp.ini’ is the main input file, and it imports the two remaining input files.

The ‘parameters.ini’ file is used to set the variable input parameters defined for the TH(O)RP protocol. These settings range from which debugging output to enable, to message generation intervals and which specialist methods to incorporate. Specialist methods includes methods to achieve fast recovery from network errors. Other settings regarding protocol parameters, from the maximum forwarding jitter to table hold times and the number of retransmissions for lost packets, can be defined here. Using the ‘parameters.ini’ file, the behaviour of the TH(O)RP protocol can be finely tuned, and it can be tested under various conditions. These variable input parameters were extensively used to optimise the default parameters for the TH(O)RP protocol.

The final input file, ‘layout.ini’, is used to define a simulation scenario. With this input file, the number of nodes to use in the simulation and their positions in the network, can be defined. The node positions can be manually set, or OMNET++ can randomly position the nodes in the network. Shut down, or power-up times for the nodes, can also be set. This was used to test the responsiveness of the TH(O)RP protocol to network changes. We can also specify the parameters of the physical layer from this file, extending the transmission power of the 802.11 radios, or setting the losses incurred during transmissions over the air. Furthermore, a channel drop rate can be specified, which tells the TH(O)RP protocol to drop a certain percentage of incoming and outgoing traffic. This was used to simulate a lossy channel environment, and was intended to measure the effect of packet losses on the TH(O)RP protocol ³.

6.2.10 Summary

The OMNET++ and mobility framework were used to simulate the TH(O)RP protocol. With the help of visual aids, network layer and application layer output, a graphical representation of the resulting tree hierarchy and statistical information, the correct operation of TH(O)RP can be verified, and its efficiency measured. It also enables us to optimise the parameters, specifically timing parameters, of the TH(O)RP protocol. With the help of the simulation results, pitfalls in the protocol design can be identified, and improvements can be implemented more effectively.

In the following section, the various tests constructed to test and optimise the components of the protocol, are discussed.

³Because an 802.11 physical layer is used, collisions and interference can occur in the communications medium, causing packets to be lost or corrupted, in addition to the channel drop-rate specified in the ‘layout.ini’ file.

6.3 Simulation scenarios

6.3.1 Overview

In order to test the different components of the TH(O)RP protocol, five different simulation scenarios were devised. The different scenarios with their intended goals are listed in Table 6.1⁴. These simulation scenarios were designed to test various aspects of the TH(O)RP protocol.

The first simulation scenario was used to compare latencies measured during simulation with those approximated by the latency models developed in Chapter 5. For conducting a large number of tests, the smaller scenario, scenario two, was used as the simulation slows down as more nodes are included in the network. Due to the latency requirement specified for the project, an estimate of the number of hops, over which a 1 second maximum latency can be guaranteed, is required. To conduct this test the third scenario was used. The system was also required to be robust and to use alternative routes if the route in use is broken or deteriorates in quality. For these tests the fourth scenario was used. Due to the large size of the proposed network, which is defined by the number of TICs connected via the network, TH(O)RP is required to scale to networks with large numbers of nodes. For these tests the fifth scenario is used. The maximum attempted number of nodes in the simulation was 225, but the simulation crashed after a few minutes, due to memory constraints on the test machine. Theoretically, the TH(O)RP protocol should scale to networks with thousands of nodes, provided the nodes high up in the tree hierarchy have sufficient storage for the information repositories. This could, however, not be tested in simulation, due to the constraint mentioned above.

The different tests were conducted in the order of appearance as indicated in Table 6.1 and optimisations or corrections made in the previous tests, were carried over into the subsequent tests. The benefits of some of these optimisations became immediately evident in the subsequent tests, whilst others had unpredicted effects when used alone. The unpredicted effects were experienced, when various methods designed to increase the responsiveness of TH(O)RP to network disturbances, were included. While some of them could function alone, some caused the TH(O)RP protocol to become unable to recover from hierarchy failures. These are discussed later.

Table 6.1: *Simulation scenarios and objectives*

Number of nodes	Node placement	Scenario objectives
6 nodes	manual	Used to compare latencies measured during simulations with those approximated by latency models.
10 nodes	manual	Test basic protocol operation and optimise timing parameters.
20 nodes	manual	Test effect of multiple hops on throughput and latency.
36 nodes	manual	Test protocol responsiveness to disturbances and represents a scaled version of a ‘real-life’ implementation.
100 nodes	random	Test scaling ability of protocol.

⁴Note that the number of nodes in Table 6.1 refer to the number of outstation nodes, as the instation is always present in the network.

6.3.2 Scenario network layout

The five scenarios as mentioned above all have specific goals that they set out to achieve, as mentioned in the previous table. The first scenario with 6 nodes follows the form of the latency models and can be seen in Figure 6.11. The network layout for the 10 node scenario is the example simulation network, as seen in Figure 6.1. The 20 node scenario, is in the form of a snake, with the instation node at the head and the outstation nodes making up the body and can be seen in Figure 6.12. The 36 node scenario, is in the form of a square matrix, with the instation at the centre and the outstations placed around the centre. This scenario can be seen in Figure 6.13(a). The 100 node scenario, is a large network with random node placements, as seen in Figure 6.13(b).

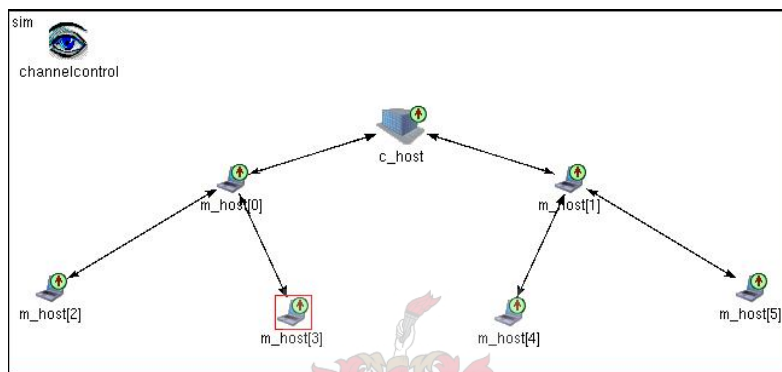


Figure 6.11: *Latency comparison simulation scenario*

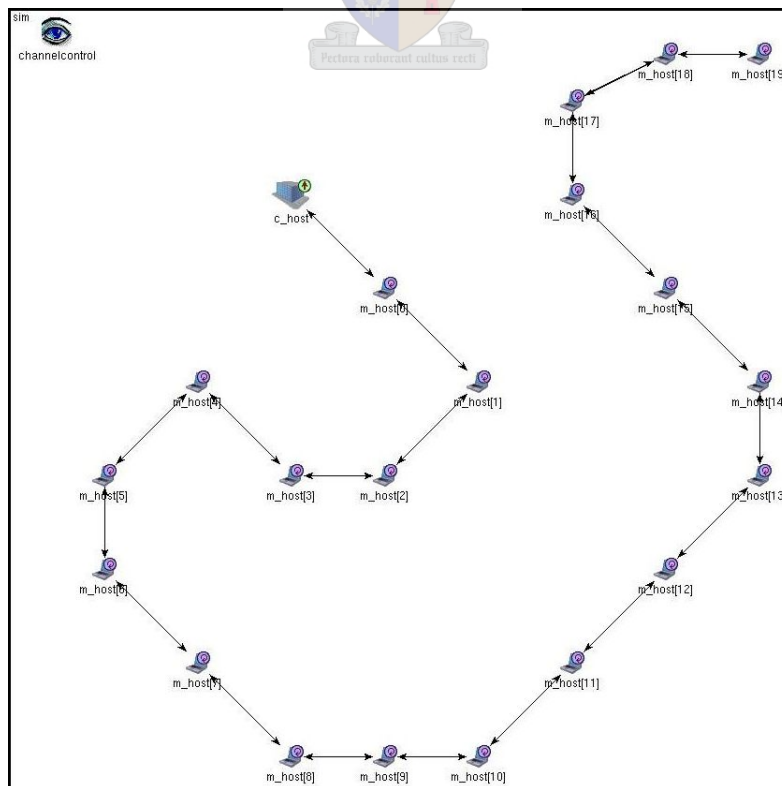
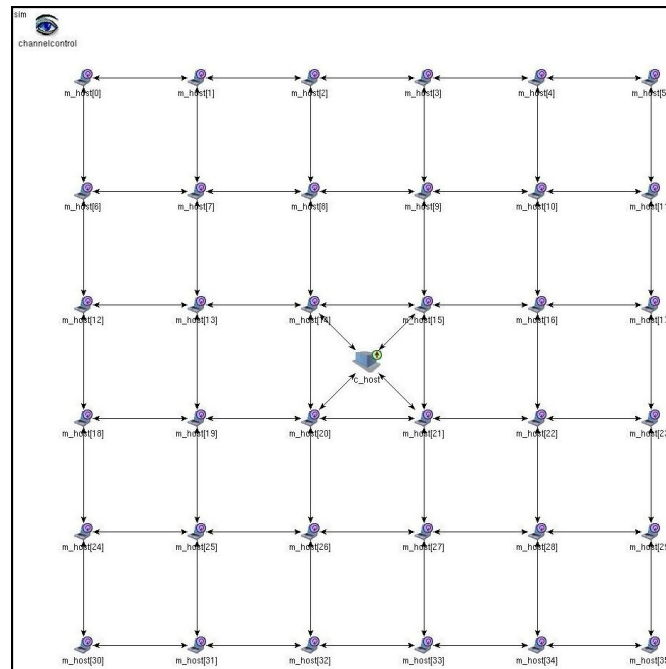
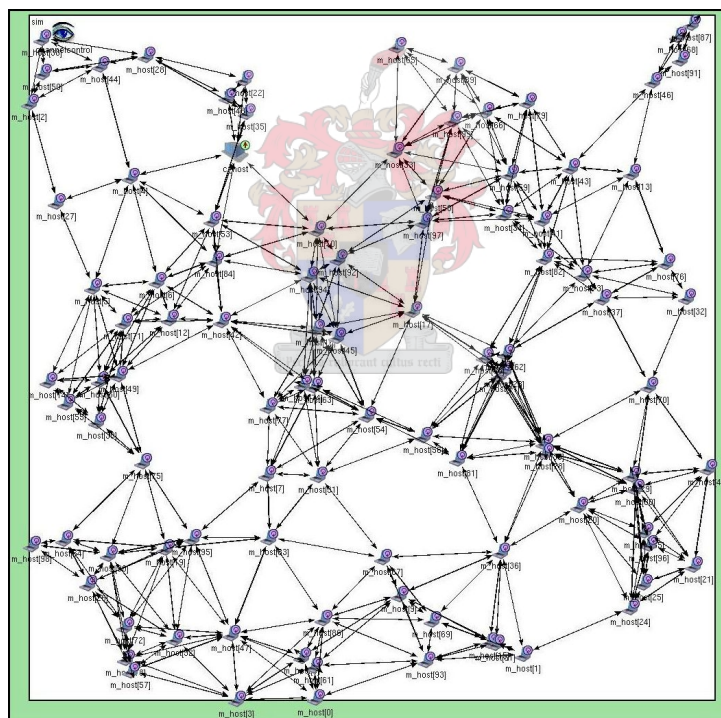


Figure 6.12: *20 Node scenario*



(a) 36 Node scenario



(b) 100 Node scenario

Figure 6.13: *The 36 and 100 node scenarios*

6.3.3 Summary

In this section, the different simulation scenarios designed to test the various aspects of the TH(O)RP protocol, were discussed. These tests were designed with a specific focus on individual aspects of the routing protocol, in order to provide the most accurate representation possible, of its behaviour. In the following chapter, these tests are conducted, in the relevant simulation scenarios, and the results analysed.

Chapter 7

TH(O)RP simulation results

7.1 Initial simulation results

7.1.1 Overview

In the initial simulations, the latencies collected during simulation, are compared to those predicted by the latency models of Chapter 5. Furthermore, the initial simulations were aimed at optimising the default timing parameters of the TH(O)RP protocol. It also aims to get a measure for the limitations of the protocol and under what conditions we can guarantee the 1 second maximum latency requirement for TRAFX messages. In this chapter, the simulation latencies are compared with the latency models. The effect of adjusting the timing parameters of the TH(O)RP protocol are tested, with simulation scenario two, as described in the previous chapter. The effect of multiple hops, on the success rate and response time of the TRAFX messages, is measured with simulation scenario three. The responsiveness of the TH(O)RP protocol, to disruptions in the network in the form of node failures, is tested with simulation scenario four. Finally, with simulation scenario five, an attempt is made to prove that the TH(O)RP protocol can scale to large networks, and that changes in the network topology are limited to a localised area.

The performance of the protocol is measured in the throughput success rate of TRAFX messages, and in the latencies incurred by those messages. All of the latency results were obtained using the mean of the measured latencies. The algorithm used to calculate the mean latencies caused an initial step response in the resulting output graphs, because of the small number of samples initially used. As more samples are included in the mean the graph converges to the real mean.

7.1.2 Comparison against latency model results

In this test, the latencies recorded during simulations, are compared with those calculated using the latency models derived in Chapter 5.6. The simulation scenario in Figure 6.11 was used and certain nodes were deactivated to get the desired situation, based on the latency models. For example, to simulate the N:1 model all the lower most nodes were deactivated. See Table 7.1, for a summary of different latencies from the models and simulations, with a Wi-Fi network interface. For the collected simulation results, the mean of the latencies obtained from the ‘omnetpp.vec’ output file, are used.

Table 7.1: Comparisons of latencies for Wi-Fi (model vs. simulations)

Model	Latency calculator 2Mb/s 0.5s jitter	Latency calculator 2Mb/s 0.125s jitter	Latency calculator 1Mb/s 0.5s jitter	Simulation 2Mb/s 0.5s jitter	Simulation 2Mb/s 0.125s jitter	Simulation 1Mb/s 0.5s jitter
1:1	0.5021s	0.1271085s	0.5042s	0.331564s	0.083594s	0.332507s
N:1	0.5021s	0.1271157s	0.5043s	0.332707s	0.0848604s	0.333507s
1:1:1	1.0055s	0.2555462s	1.0111s	0.547317s	0.140404s	0.548574s
N:N:1	1.0087s	0.2586934s	1.0180s	0.629092s	0.158144s	0.631397s
Retry	0.5023s	0.1273201s	0.5047s	0.746538s	0.194756s	0.813343s

Due to simplifications and assumptions made in the development of the latency models, the calculated latencies tend to reflect a worst case scenario, and tend to be longer than the results collected in the simulations. This is due to the fact that the single hop models adds an additional forwarding jitter, which is excessive, whereas the simulations do not add this additional forwarding jitter. For the retransmission models, this is, however, not the case, where the latency model approximated a lower latency than those measured during simulations. This is due to the way in which retransmissions are handled. Retransmitted messages are subjected to a retry timer, and after the timer expires the message is again subjected to the forwarding jitter before being output on the network interface. This is somewhat inefficient and could be optimised in future implementations of TH(O)RP.

While the latencies calculated with the latency models do not exactly match those of the simulations, they do give an estimation of what the latencies in the simulations would be. They also reflect the impact that the forwarding jitter and multiple hops have on the latencies. In addition to this, they also provided a means of deriving approximately how much bandwidth is required to process the amount of information being generated by TH(O)RP. For example, some of the results for technologies that had bandwidths of below approximately 500kb/s, became unstable with the N:N:1 model. Also, GPRS, which had the lowest device bandwidth, became unstable with a multitude of models (See Table 2.6).

7.1.3 Optimising TH(O)RP timing- and retry parameters

The timing parameters of the TH(O)RP protocol can be divided into three categories, namely message generation intervals, forwarding jitter and table hold times, specifically the duplicate set hold time. The retry parameters optimised, are the retry table timeouts and the number of retransmissions per message. It should be noted, that all of these tests were conducted in a simulation environment where all the links were subjected to a 10% drop rate, so 1 out of 10 messages are dropped to simulate message losses in the network.

Message generation parameters

The TH(O)RP messages can be divided into time critical and non-time critical classes. The MID and HNA messages are non-time critical and do not effect the performance of the protocol in anyway, as they provide auxiliary functionality to the protocol. The HELLO, ROUTE, DATA and ACK messages are time critical and provide the primary functionality of the TH(O)RP protocol, so only their timing parameters will be optimised.

The first test conducted, was to modify the default HELLO generation rate, which in turn effect the ROUTE update- and parent selection intervals. More precisely, the parent selection and ROUTE update interval is taken as three times the HELLO generation interval. These parameters were optimised to give the highest throughput success rate, whilst offering the lowest latency possible. The different parameters and the resulting effects are summarised in Table 7.2 and Figure 7.1. The resulting parameters that investigated, are the throughput success rate, the data to control traffic ratio and the latency of the TRAFX messages.

What was evident from these results, is that the optimisation process became a trade-off between latency, the throughput success rate and the data to control traffic ratio. When the HELLO message generation rate was increased, a considerable amount more control traffic was created, with which data traffic needed to contend. Decreasing the HELLO message generation rate, caused the protocol to become extremely slow to react to network topology changes, and a lost route update would cause the protocol to experience long periods of silence as the ROOT is no longer aware of it. This resulted in a decrease in throughput success rate and the poor response can be seen in Figure 7.1, in the form of a jigsaw anomaly in the response time of the final two tests. This anomaly is not as apparent in the other three cases, as routes tend to stay more stable.

After the test the default HELLO message generation interval was set to 2s, as this achieved the best compromise between the throughput success rate and response time. In addition to optimising for time, it was also desirable to keep the amount of control traffic in the network low. This was measured in the data to control traffic ratio, and the default generation interval produced a high ratio. However, just optimising for an optimal ratio, provides unsatisfactory results in terms of the latency and throughput success rate, which was shown by the final two cases.

In conclusion, the 2s HELLO message generation, provides the best trade-off between the three measurement criteria.

Table 7.2: HELLO message optimisation summary (default marked with *)

HELLO interval	Parent select and ROUTE interval	Success rate	Data to control traffic ratio
0.5s	1.5s	90.35822%	0.9840
1s	3s	90.79978%	1.9877
2s *	6s	91.38472%	2.8906
4s	12s	87.01557%	3.1116
8s	24s	85.62231%	3.1702

Forwarding jitter

In any communication system using a shared communication medium, there will always be collisions between users of the communications medium trying to access it simultaneously. In order to decrease the number of collisions, a forwarding jitter was incorporated into the TH(O)RP protocol. The forwarding jitter is a short, random wait-time applied to all messages, before being output on the network interface.

While the forwarding jitter has a beneficial effect by reducing collisions, it has the negative effect of introducing a wait time at every single node for outputting messages into the network.

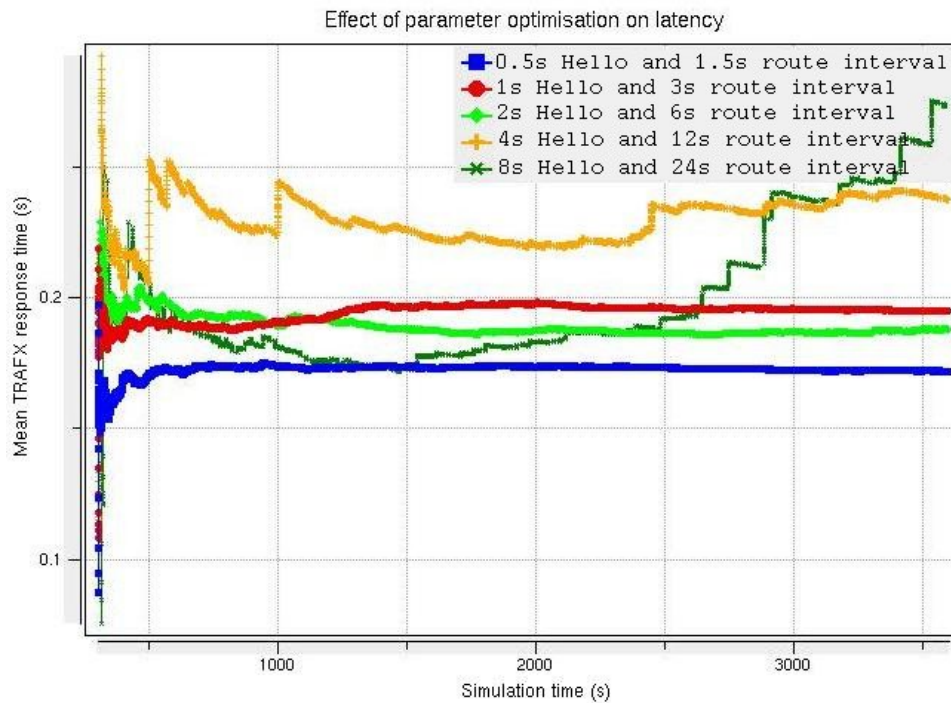


Figure 7.1: HELLO message optimisation effect on latency

Table 7.3: Maximum jitter optimisation summary (default marked with *)

Maximum jitter	Success rate
0.5s	93.61702%
0.25s	93.57228%
0.125s *	93.22755%
0.0625s	91.61896%

This adverse effect becomes apparent, when messages are routed over a large number of hops, and it eventually becomes a limiting factor in the maximum number of hops, over which a 1 second maximum latency can be guaranteed.

The default value for the maximum forwarding jitter, was initially the HELLO generation interval divided by 4. This amounted to a maximum forwarding jitter period of 0.5 seconds. The forwarding jitter used by a node is calculated as a random number between $0 \dots jitter_{max}$ and is re-calculated every time a node outputs a message. In Table 7.3, it can be seen how the maximum forwarding jitter can affect the throughput success rate, if it is made too small. In Figure 7.2, it can be seen how the overall latency can be reduced, by decreasing the maximum forwarding jitter parameter. The four values for the jitter was 0.5s, 0.25s, 0.125s and 0.0625s.

From the results of the test, it can be seen that latency can be drastically lowered, by decreasing the maximum forwarding jitter. However, when the forwarding jitter is decreased lower than 0.125s, a drop in the throughput success rate is experienced, due to more collisions in the shared communication medium. From these results, the new default for the maximum forwarding jitter was set to 0.125s, as this provided a substantial decrease in latency whilst keeping the throughput success rate high.

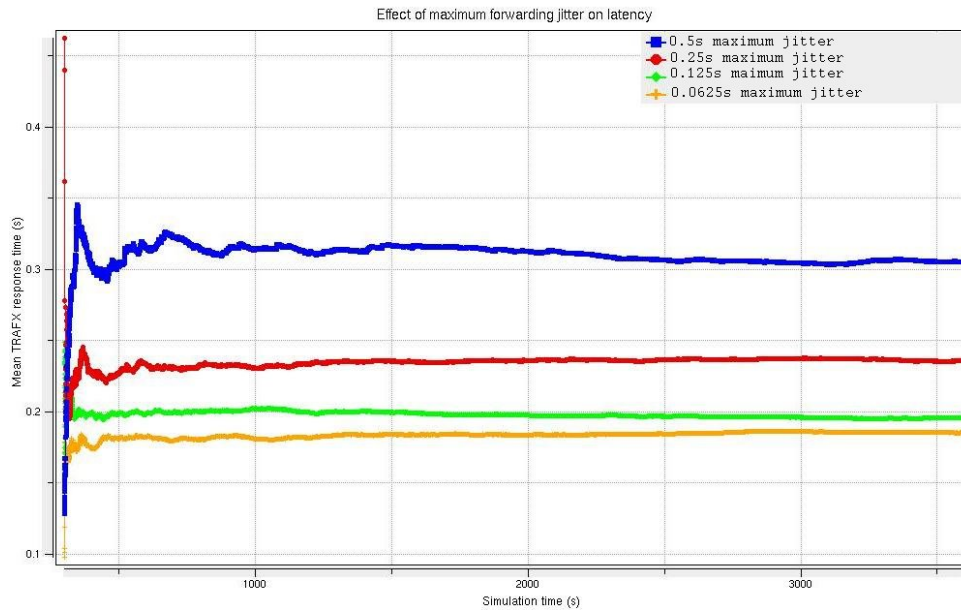


Figure 7.2: *Effect of maximum forwarding jitter on latency*

Table hold times and time outs

All of the information repositories in the TH(O)RP protocol have timed entries, meaning that they are kept for a certain period of time, and removed if they are not refreshed within a certain time frame. This time frame is defined by the hold times of the table entries.

The hold times optimised, are the duplicate set hold time and the retry time-out. The reason for optimising the duplicate set hold time, was that the duplicate set can become large. The duplicate set stores an entry for each message forwarded by a node, and nodes high up the hierarchy experience large amounts of traffic. From initial tests, it was observed that the duplicate set became large in big networks. It was decided, to reduce the hold time of the duplicate set from 30s to 10s, due to the absence of circular routes in the TH(O)RP protocol. The reason for keeping the duplicate set, was to avoid re-processing messages when routes, which could cause circular routes to form temporarily.

Another observation that was made, was that the default hashing algorithm, used in the information repositories was not providing a well enough spread amongst the indexes of the duplicate set. It was, therefore, decided to implement a more efficient hashing algorithm. The summing components hash code was chosen, with the multiply and add (MAD) compression map [35]. The result, was that the entries of the duplicate set would be spread more widely through its indexes. The resulting optimisation can be found in Table 7.4. The reason for keeping hash tables for the information repositories, is to minimise the lookup time for entries in these tables, in order to minimise the internal latencies introduced by the TH(O)RP protocol. With the simulation of larger networks, it was noted that the route table also grew considerably, especially for nodes high up in the tree hierarchy. Therefore, it was decided to also use the MAD hash for the route table.

The retry timeout parameter, determines the amount of time the protocol waits for a message to be acknowledged, before it assumes that a message was lost. It was important to define this parameter correctly, as it has an effect on the overall latency and success rate of the network. If

Table 7.4: *Effect of optimised hash algorithm on duplicate set storage*

Hashing technique	Number of entries	Indexes used	Average number of entries per index
Default (cast to short)	95	3	31.67
MAD	95	7	13.57

Table 7.5: *Effect of retry time on success rate (default marked *)*

Retry time	Success rate
$0.5 \cdot jitter_{max}$	91.61896%
$1 \cdot jitter_{max}$	93.16339%
$1.5 \cdot jitter_{max}$ *	93.22755%
$2 \cdot jitter_{max}$	93.57228%

the parameter is too short, the acknowledgement or the original message, can still be in transit to the destination, when a message is registered as lost. This would have the effect of reducing the efficiency of the retransmission effort, as it will waste allowed retransmissions on messages that have actually been successfully received. If the parameter is too long, the maximum latency of the network will increase, as it will take a long time before a message loss is registered, which causes long wait-times before a lost message is retransmitted.

The reference that the retry time is measured against, is the RTT of a message between neighbouring nodes. Due to the speed of the links, at 2Mb/s, the major factor in the RTT, is the maximum forwarding jitter. It was therefore decided, to set the retry time according to the maximum forwarding jitter. The candidate retry times were: $0.5 \cdot jitter_{max}$, $jitter_{max}$, $1.5 \cdot jitter_{max}$ and $2 \cdot jitter_{max}$. The simulation results are summarised in Table 7.5, and the resulting latencies are displayed in Figure 7.3.

As expected, the test where the retry time was lower than the maximum jitter, a lower success rate was observed, compared to the other three cases, where the retry time is greater or equal to the jitter. This can be attributed to retransmissions wasted on successfully delivered messages. The optimum retry time, was $1.5 \cdot jitter_{max}$, as it provided a good success rate and kept the latency low.

It should be noted, every time a message is retransmitted, the retry wait time is added to the latency of that message. This has the effect, of increasing the average latency of messages in networks, or along routes with poor link qualities, causing more messages to be dropped. This effect can be seen in Figure 7.4, where a test was conducted with the new default retry time, in a network where the drop-rate of the channels was increased from 10% to 40%. In this test, the mean latency has increased from approximately 0.22s to 0.4s for the case with $1.5 \cdot max$ jitter. This is almost double the original, which confirms that the overall quality of the communication links in the network, will have a big effect on the latency of the network.

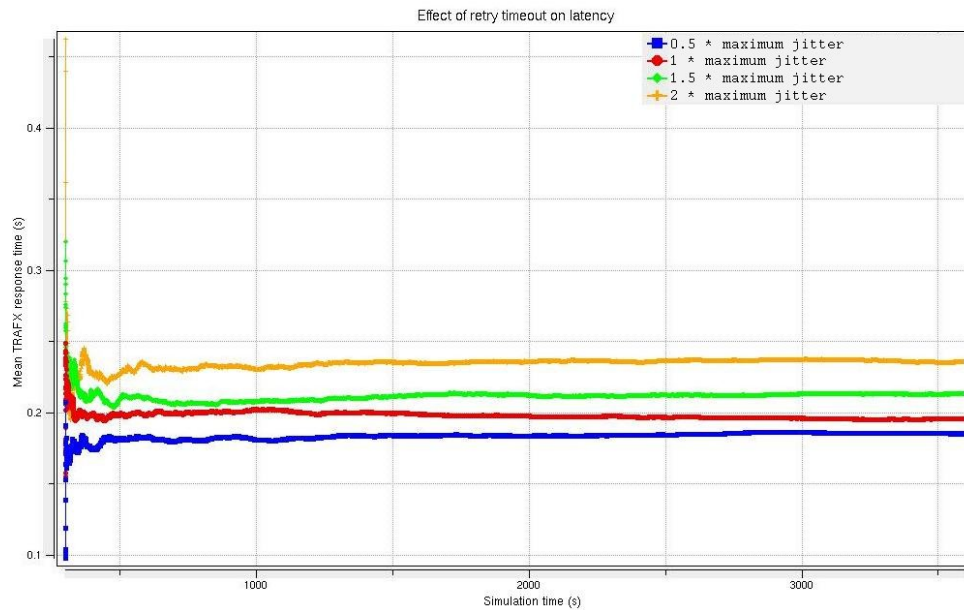


Figure 7.3: *Effect of retry timeout on latency*

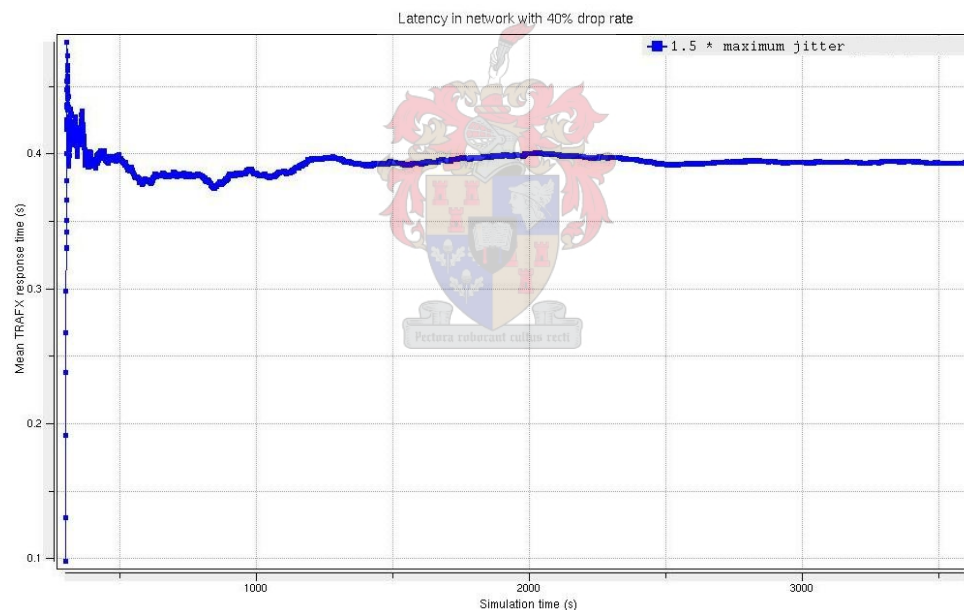


Figure 7.4: *Effect of poor link qualities and retransmissions on latency*

Number of retransmission attempts

The delivery system of the TH(O)RP protocol, is based on a best effort system, with multiple retransmissions allowed per message per hop. This provides a limited ability, to recover from errors in the communication medium. Although more efficient and reliable delivery systems exist, the most popular being the TCP protocol, it was decided to go with a simple delivery system. This is due to the congestion control and message throttling mechanisms in the traditional TCP, which tend to fail over multi-hop wireless networks, reducing the overall throughput. Various flavours of the TCP protocol are being developed for use in wireless multi-hop networks, and they can be considered as a replacement delivery system in the future.

Table 7.6: *Effect of number of retransmissions on success rate (default marked with *)*

Number of retries	Success rate
0	32.06436%
1	43.04871%
2	68.02418%
3	78.52553%
4 *	81.19478%
5	81.60528%
6	81.39842%
7	81.20375%
8	80.30937%

In the delivery system implemented, the number of retransmissions allowed per message per hop, can be adjusted. By decreasing this number, the protocol is inhibited from effectively managing lossy networks. Increasing this number by to great a margin, tends to congest the network with old messages, decreasing the available bandwidth for newer messages. This became apparent in the tests conducted. When the number of retransmissions were increased past a certain threshold, a decrease in the throughput success rate was observed. Take note, the messages that are eligible for retransmission, are DATA messages and ROUTE messages. Thus, increasing the number of retransmissions, not only provides better success rates, but also more stable tree hierarchies.

From the tests in the previous section, it was observed that the retry time and number of retransmissions required to get a message through the network, could severely affect the latency of the network. This was again reflected in these tests, as the latency of the network increased as the number of retransmissions increased. Note that these tests were executed with a 30% channel drop-rate, to emphasise the effect of packet losses and retransmissions. The results for the optimisation on the number of retransmissions, are summarised in Table 7.6, and the resulting latencies can be seen in Figure 7.5.

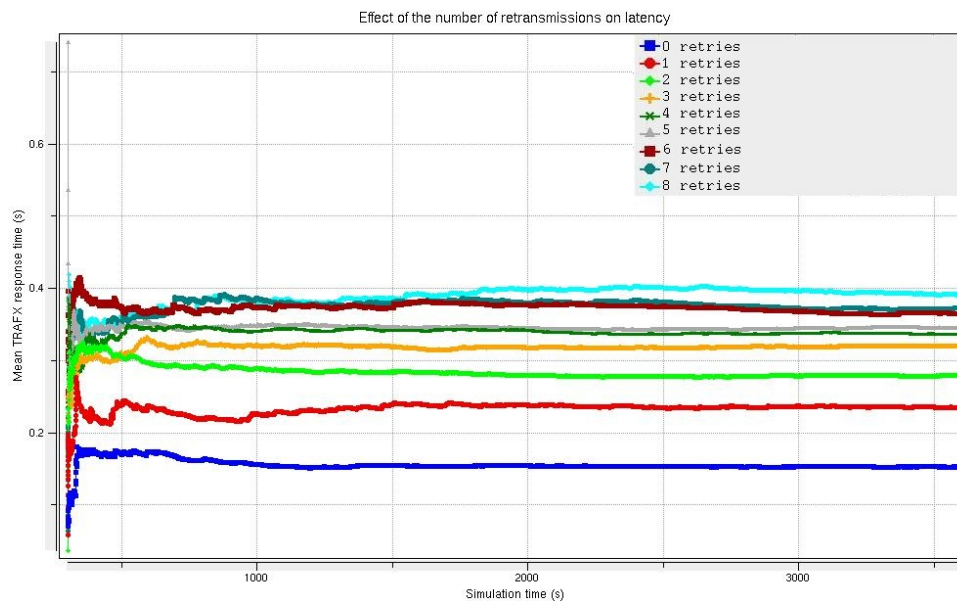
**Figure 7.5:** *Effect of the number of retransmissions on latency*

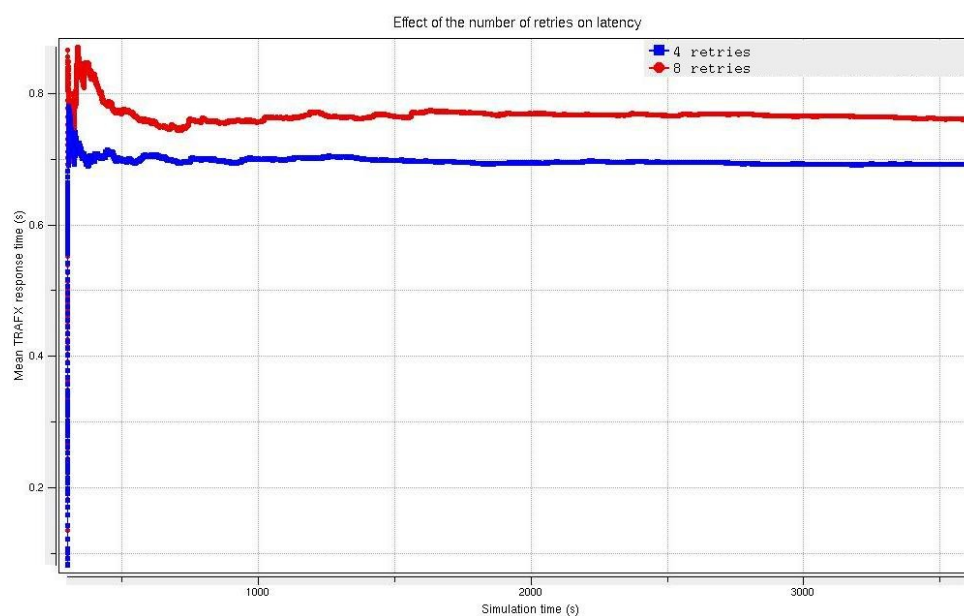
Table 7.7: *Effect of number of retransmissions on success rate, a second test*

Number of retries	Success rate
4	50.34162%
8	46.01070%

From the results obtained, it can be seen that the success rate increases as the number of retransmissions allowed increases, but the average latency also increases. A peak is reached when the number of retransmissions is set between 4 and 7, and a drop-off in success rate is experienced, when the number is set to 8 retransmissions. This prompted the use of 4 retransmissions as the new default, as increasing the number of retransmissions past 4 did not provide any substantial increase in the success rate, and only introduced a higher latency.

The full impact of retransmissions on the network latency and throughput, cannot be fully appreciated in a small network environment, such as simulation scenario 2. In order to better illustrate this effect, two more tests were conducted with the 100 node simulation scenario, with a channel drop rate of 30%, and with 4 and 8 retransmissions respectively. In a big network, the messages have longer routes to travel, and the chances of old messages traversing the network while new ones are introduced, is increased. The results are summarised in Table 7.7 and Figure 7.6. As expected, the results for the 8 retransmissions deteriorates the success rate, as it caused the network to become congested with old messages.

Because the delivery system is a best effort system, successful message delivery cannot be guaranteed under all conditions. This stresses the importance, of implementing a network environment with links of the highest quality possible, which can be achieved through strategic antenna placement in the final implementation. It should be remembered that the delivery system implemented in TH(O)RP, only aims to increase the stability of the network. In order to truthfully guarantee reliable application layer message delivery, a transport protocol needs to be used that can guarantee message delivery.

**Figure 7.6:** *Effect of the number of retransmissions on latency, a second test*

7.1.4 Improved acknowledgement and retransmission cooperation

During tests with a large number of retransmissions available, for example 8 retransmissions, it was found that the implied acknowledgement system contained a flaw. The implied acknowledgement system, depends on the fact that a forwarded message is successfully overheard at a node waiting for an acknowledgement. The problem is that implied acknowledgements could get lost, in network environments with lossy channels.

This caused nodes waiting for an acknowledgement, to keep retransmitting their messages, because they did not receive an acknowledgement. Also, a node that forwards a message, marks it as previously processed, and every time that message is retransmitted due to a failed implied acknowledgement, it will ignore the message. It also does not forward the message again, which means that the implied acknowledgement does not occur. In Figure 7.7, the original successful acknowledgement, a failed acknowledgement and the new improved acknowledgement system can be seen.

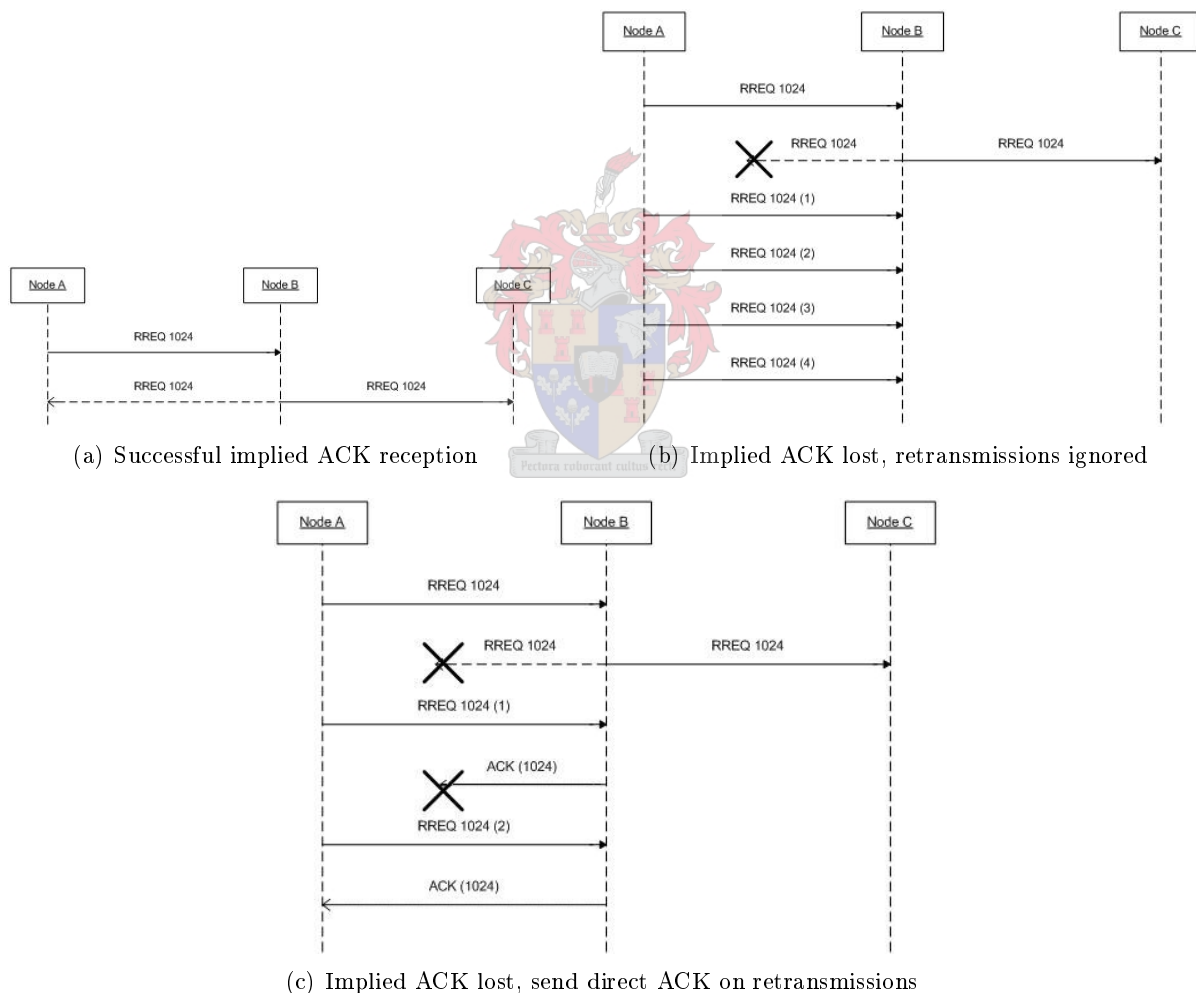


Figure 7.7: *Different acknowledgement scenarios*

A test was conducted, in a highly lossy environment (50% channel drop rate), and with 10 allowed retransmissions per message to illustrate the above mentioned problem. This introduced the problem that messages successfully received and forwarded, were not received as acknowledged by the originating node, which would then continue retransmitting the message until the

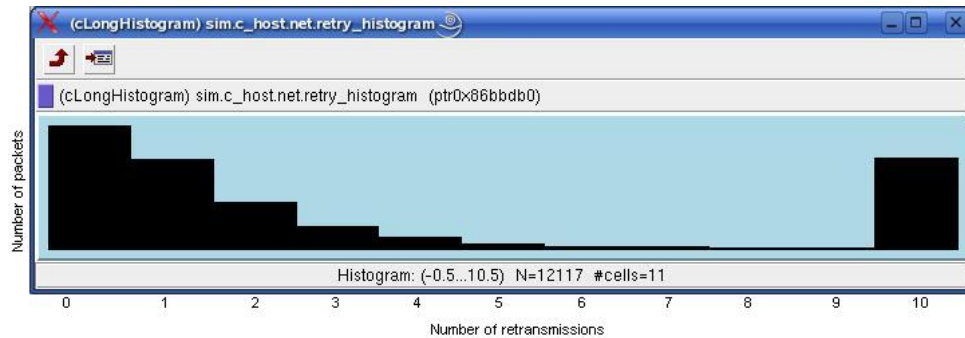


Figure 7.8: *Histogram of retransmission distribution*

maximum number of retransmissions was reached. This fact is clearly illustrated in Figure 7.8, where a histogram of the average number of retransmissions at nodes in the network, can be seen. This histogram was observed at the instation. As can be seen, there is a fair distribution between the first two cells, and then a large number of entries in the final cell. It would be expected for the distribution, to be more evenly spread along the different number of retransmissions available, but they are concentrated in the first two and the maximum number of retransmissions allowed cells respectively. This indicates that a large number of messages were retransmitted the maximum number of times allowed, which is undesirable, due to the large amount of unnecessary control traffic in the network.

The solution to the problem, was to send an acknowledgement to the source every time a previously processed and forwarded message is received, instead of ignoring it. This caused a drastic reduction in the amount of control traffic present in the network. The retransmission histogram was also more evenly distributed, with considerably less messages being retransmitted for the maximum number allowed. See Figure 7.9, for the improved histogram distribution.

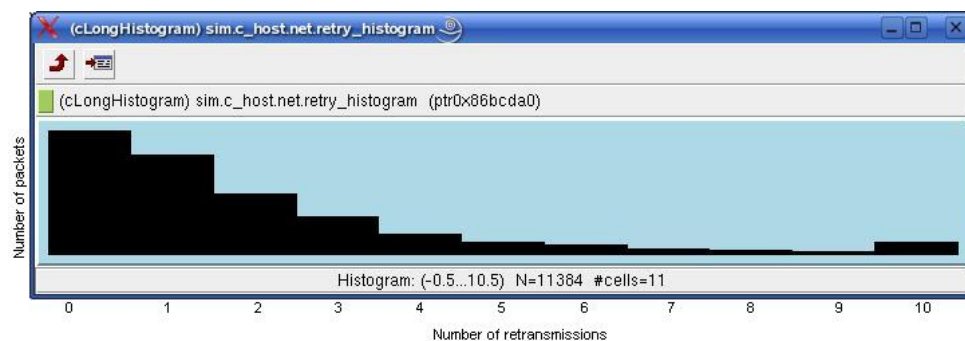


Figure 7.9: *Improved histogram of retransmission distribution*

7.1.5 The effect of communications data rate on throughput and latency

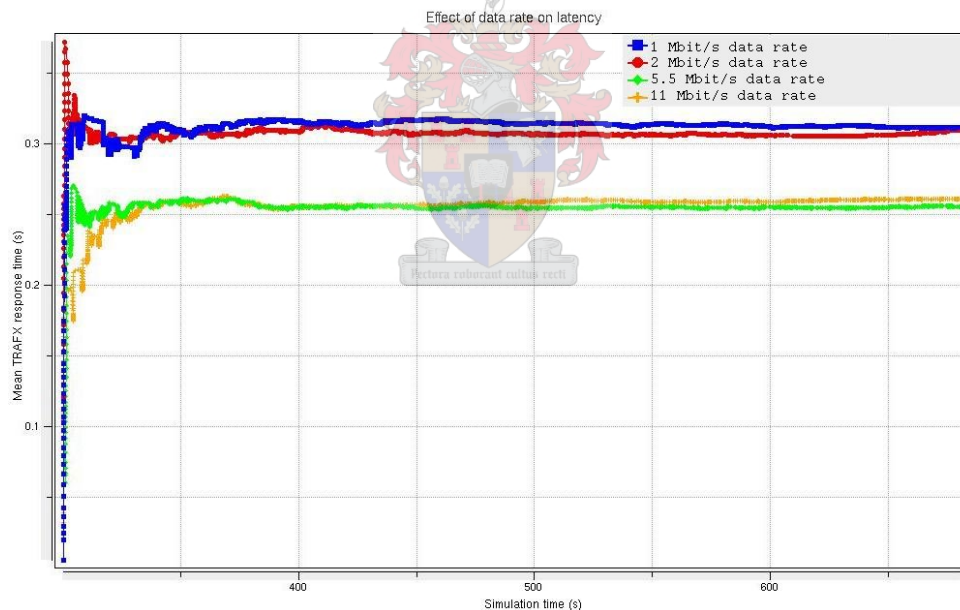
All of the simulations, were conducted with an 802.11 network interface running at 2 Mb/s. A test was conducted, to measure the affect of the data rate on the success rate and latency of the network. While the biggest contributor to the network latency has been the wait times introduced by TH(O)RP, there is also a latency introduced by the data rate, which determines the amount of time it takes to transmit information over the air. Only valid 802.11 data rates

Table 7.8: *Effect of data rate on success rate*

Data rate (Mb/s)	Success rate
1	98.31458%
2	98.86061%
5.5	99.37062%
11	99.40797%

were allowed by the mobility framework, namely 1Mb/s, 2Mb/s, 5.5Mb/s and 11Mb/s. These tests were conducted for an ideal network environment, with a 0% channel drop rate and on simulation scenario four.

The results of this test are summarised in Table 7.8, and the resulting latencies are seen in Figure 7.10. As can be seen from the results, when the data rate is increased past 2Mb/s, the latency introduced by transmission time becomes negligibly small, and no longer increases the success rate or latency of the network. This is proved by the similar latencies for the 5.5Mb/s and 11Mb/s tests. The success rate increases as the data rate increases, because message transmission times are shorter. This reduces the amount of time a message takes to traverse the network, and the number of messages contained in the network. A reduction in collisions is also obtained, because a transmitting node occupies the communication medium for a shorter time period.

**Figure 7.10:** *Effect of data rate on latency*

7.1.6 The effect of multiple hops on TH(O)RP

As was discovered in previous tests, each hop in a multi-hop network introduces its own latency. Sources of these latencies are:

- Rescheduling of messages for forwarding with a forwarding jitter.
- Message retransmission due to message losses.

- Internal look up times of information repositories.
- Data transmission times.

Due to the wait- and hold-times of TH(O)RP and the finite data rates, a limit is placed on the maximum number of hops over which a maximum latency of 1 second, can be guaranteed. Tests are conducted to determine this limit, and to test the effects of multiple hops on the success rate. These tests were conducted in line with simulation scenario three (See Figure 6.12). The test is conducted for two channel drop rates; firstly, an ideal environment is used, where only radio collisions and interference can occur (with a 0% drop rate), and secondly, a 10% drop rate is introduced.

In order to ensure that excessive control traffic is not included during these tests, only the number of nodes required to carry the message along the desired number of hops, are activated. For instance, to test the throughput over 10 hops, only the first 10 nodes are activated, with the remaining 10 deactivated. The results are summarised in Table 7.9, and Figures 7.11 and 7.12.

When comparing the results of the 0% and 10% tests, the effect of message losses and re-transmission, coupled with multiple hops, can be seen on the average latency of the network. In the first test, the 1 second maximum latency requirement is satisfied up to 11 hops, while in the second, it is only satisfied up to approximately 8 hops. The number of hops over which a 1 second maximum latency can be guaranteed, will decrease as the channel quality deteriorates. This, once again, stresses the need for a network with quality links, to ensure that the network latency is kept to a minimum.

Table 7.9: *Effect of multiple hops of throughput success rate*

Number of hops	Success rate (0%)	Success rate (10%)
1	100.0000%	100.00000%
2	100.0000%	94.90909%
3	100.0000%	88.45455%
4	100.0000%	83.30303%
5	100.0000%	77.36364%
6	100.0000%	74.24242%
7	100.0000%	66.96970%
8	99.98551%	64.60606%
9	99.98551%	62.51515%
10	99.98551%	59.93939%
11	99.98551%	55.75758%
12	99.97101%	54.36364%
13	99.97101%	51.81818%
14	99.97101%	49.84848%
15	99.95652%	47.57576%
16	99.94203%	42.36364%
17	99.92754%	39.15152%
18	99.92754%	34.90909%
19	99.92754%	33.63636%
20	99.91304%	30.06061%

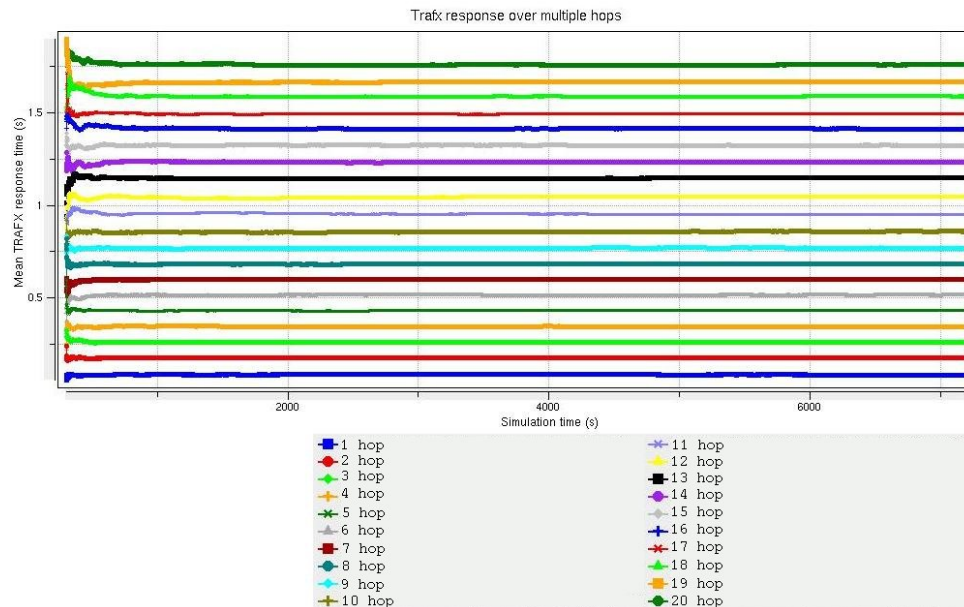


Figure 7.11: Effect of multiple hops on latency (0% lossy channels)

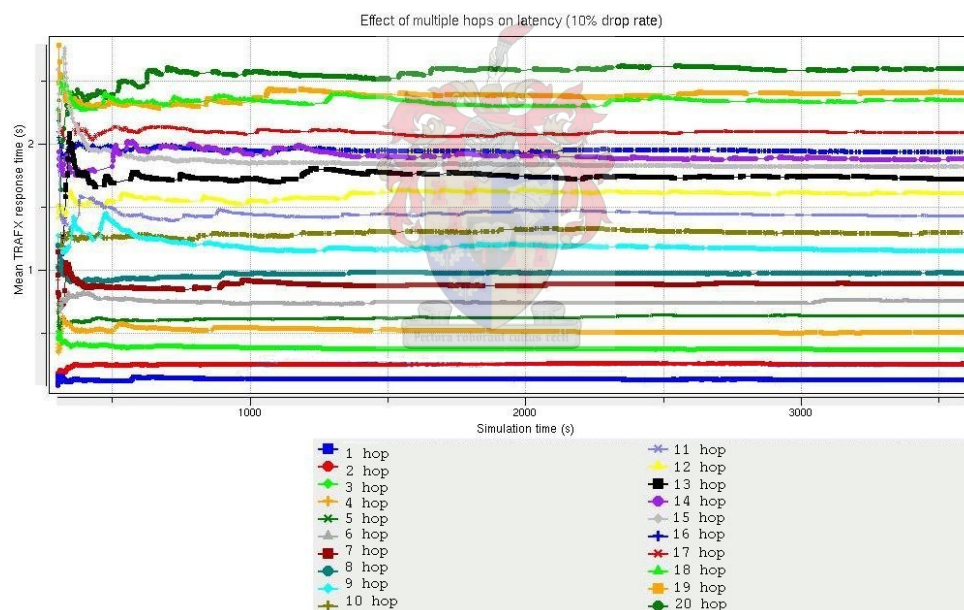


Figure 7.12: Effect of multiple hops on latency (10% lossy channels)

7.1.7 The responsiveness of TH(O)RP to route failures

As mentioned earlier, the proposed network will have static nodes, but links might fail from time to time. This will cause routes to be broken, which will need to be repaired. Because TH(O)RP maintains only a single optimal route to the instation per node, it is required to recover from route failures quickly, to minimise the losses incurred during such interruptions.

To test the responsiveness of TH(O)RP scenario four is used (See Figure 6.13(a)). This scenario resembles a smaller version of the real life system. This scenario was modeled after square city blocks, with TICs on the corners of the city blocks. In reality, the blocks of any metropolitan area are rarely ever this neat, but TH(O)RP will adapt to any network topology

it is presented with, due to the ad hoc nature of the protocol. For this example, it was assumed that a node can see four nodes, except at the centre, where they can see 5 and at the edges, where they can see either 2 or 3. In order to ensure connectivity, it can be assumed that a node will never lose connectivity to all of its neighbours, and that at least one neighbour is not a child of this node. Thus, every node will always have at least one other node, that is not a child, visible at all times during the simulations.

In order to test the responsiveness, only polylogue messages are used. These messages, are all addressed to the node in the top left hand corner of the simulation scenario in Figure 6.13(a), or `m_host[0]`. See Figure 7.13 for the resulting tree hierarchy of the matrix scenario, where `m_host[0]` is highlighted in black. The network address, or the number indicated inside the nodes of the graphical tree hierarchy representation, of `m_host[0]` is 43. From now on `m_host[0]` will be called node 43. In Figure 7.14, a trail of bubble message can be seen from the instantiation to node 43. This is useful in quickly identifying the current communications route being used. As the network is disrupted, the tree hierarchy is forced to reform, moving the location of node 43 in the hierarchy. This is graphically represented later.

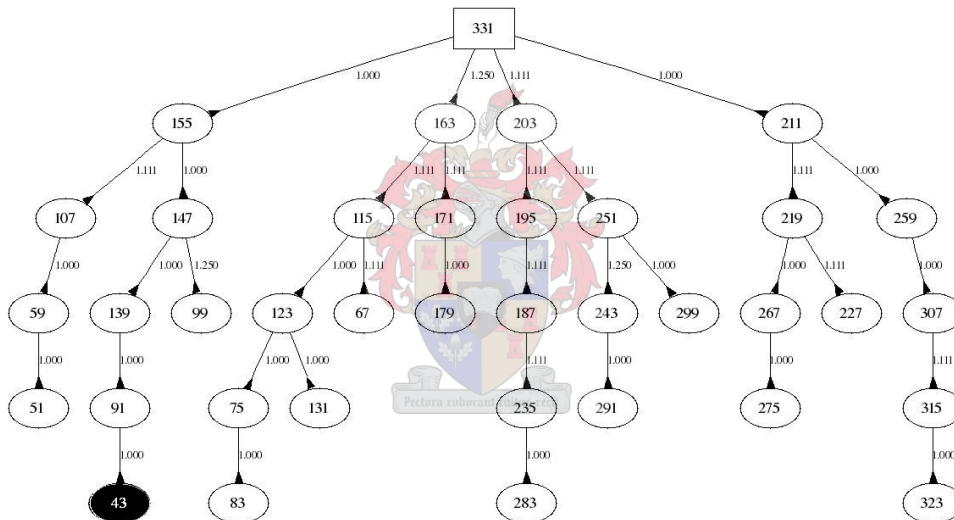


Figure 7.13: Graphical representation of matrix scenario tree hierarchy with node 43 highlighted

Five responsiveness tests were conducted, where certain nodes are switched off after 10 minutes. These included deactivating a single node at the top of the route, and a single node at the bottom of the route of node 43 to the root of the network. The tests also included deactivating all the nodes along the route of node 43, and multiple nodes around node 43. Finally, multiple nodes throughout the entire network were deactivated. In this test, the route length of node 43 was increased from 5 hops to 20 hops. The eventuality of this happening in any real life scenario, is highly unlikely, but it helped the development of methods that decrease the recovery times of TH(O)RP.

All of these tests, were conducted in an ideal simulation environment with 0% drop rate on the communication channels. Losses can still occur due to radio interference and collisions. The simulation results are summarised in Table 7.10. The newly formed tree hierarchies can be seen in Figures 7.15(a) to 7.15(e). Look closely, at how the placement of node 43 moves around, as the tree hierarchy changes to reflect the new network topology.

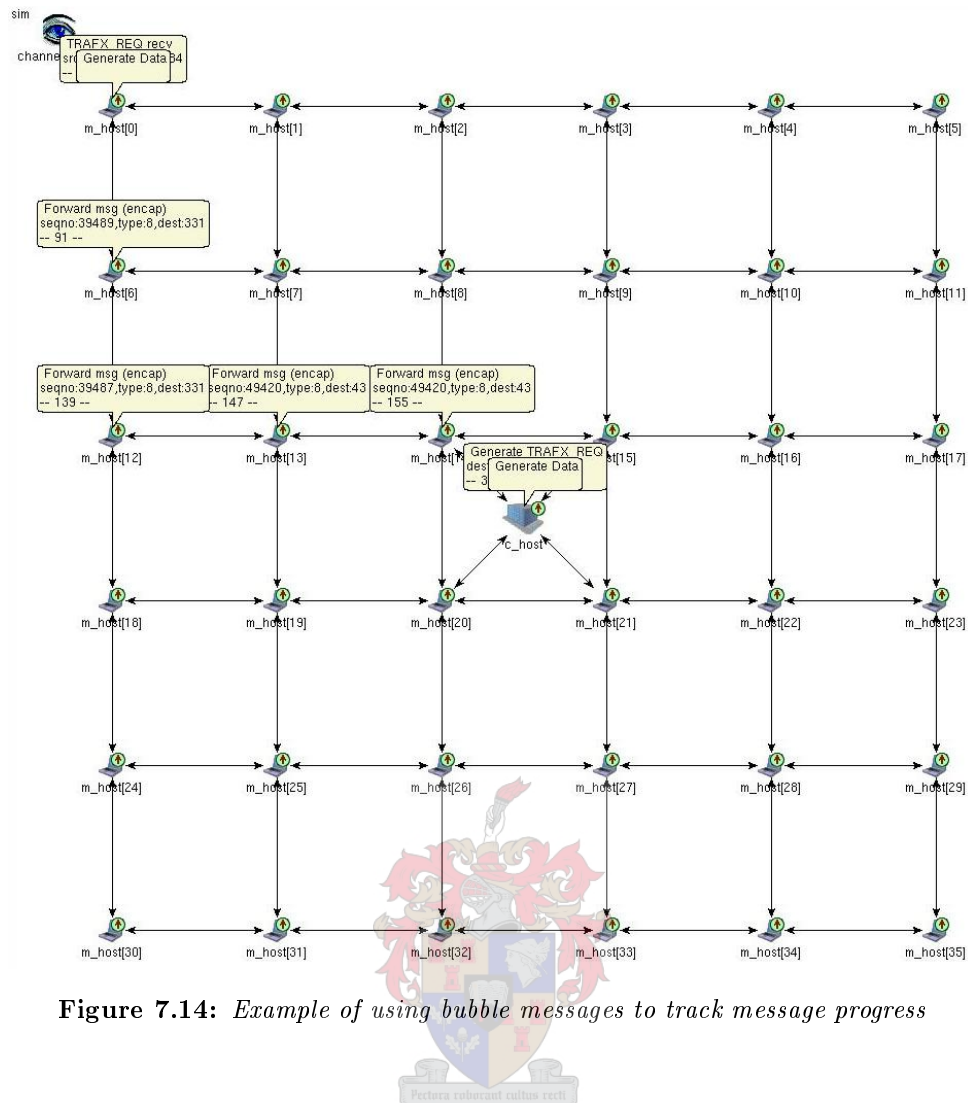


Figure 7.14: Example of using bubble messages to track message progress

Due to the periodic nature of TH(O)RP, it is naturally slow to react to and recover from node and route failures. This is because there is no active mechanism to detect route failures. As a route fails, it will deteriorate in quality, and at the next route update event TH(O)RP will pick a new route. Slow reactions were expected, but the responsiveness is slower than originally expected. Note that route recovery does not necessarily occur at the node where the route begins. Any node along the route, can set up alternative routes by choosing an alternative parent, which would remove the need for node 43 to recover at all. In these situations, node 43 would remain oblivious to changes in the route. It might just realise at the next route update interval, that the cost of the route in use has changed. In other cases where the nodes in node 43's neighbourhood are deactivated, it is required to recover from the failure by attempting to select a new parent, and updating its route through this new parent. In the final example, multiple nodes lost their routes and parents, delaying the recovery time as multiple nodes had to realise that they lost their routes. In addition to this, some of these nodes picked other nodes that have lost their routes, but have yet to change their status to orphaned, and a back and forth route recovery and route failure situation occurred. The result is that it takes a long time for all the nodes to recover their routes. In Figure 7.16, multiple nodes were deactivated, and in Figure 7.17, a new route was established, and messages are being passed along the new route. The resulting responsiveness of TH(O)RP was rather disappointing, especially in view of the maximum 1 second latency. In Section 7.2 methods were developed to speed up recovery of failed routes.

Table 7.10: Route recovery times

Test	Approximate recovery time (s)
1	12
2	12
3	18
4	18
5	126

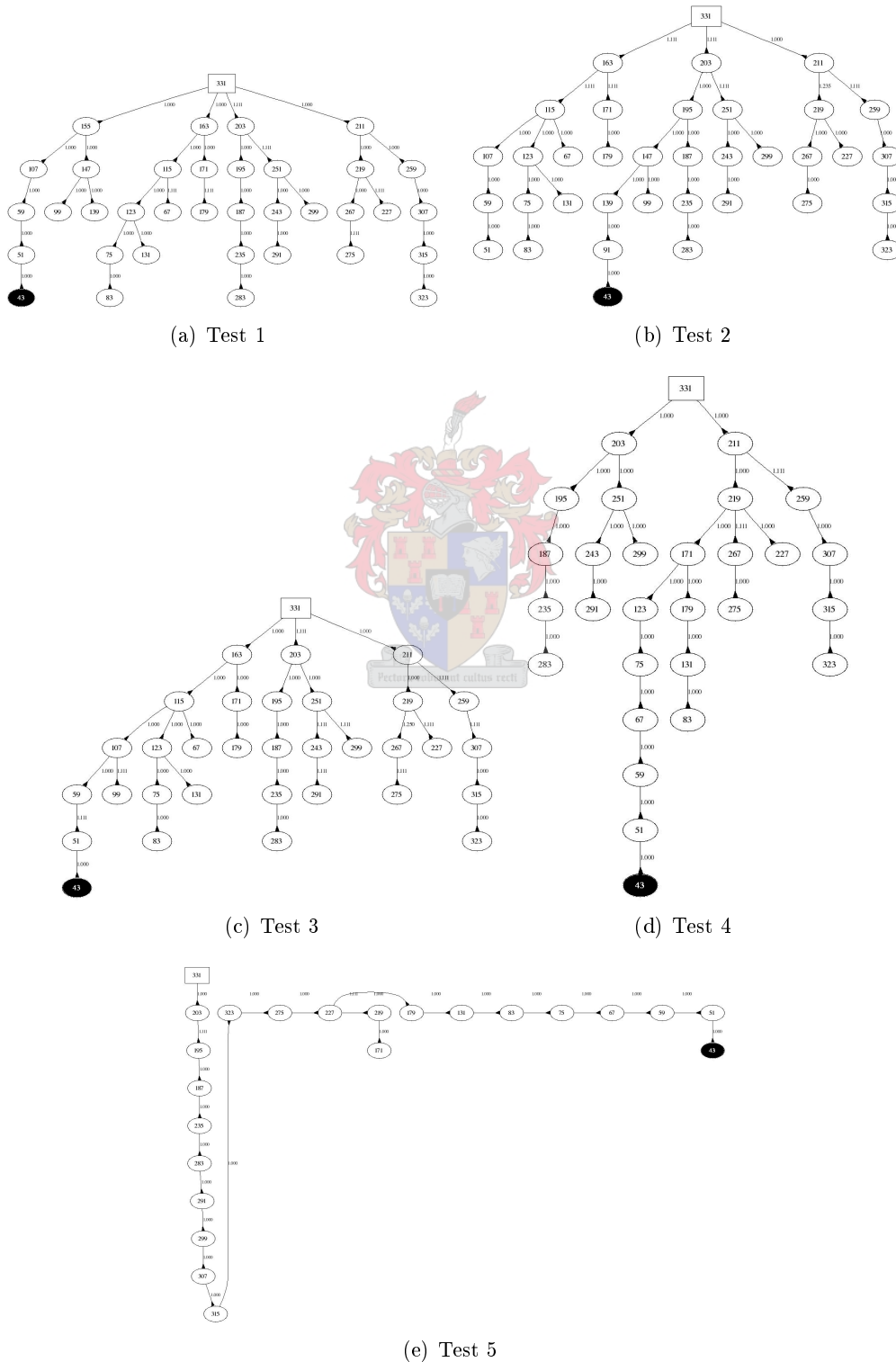


Figure 7.15: Tree hierarchy graphical representation of recovery tests

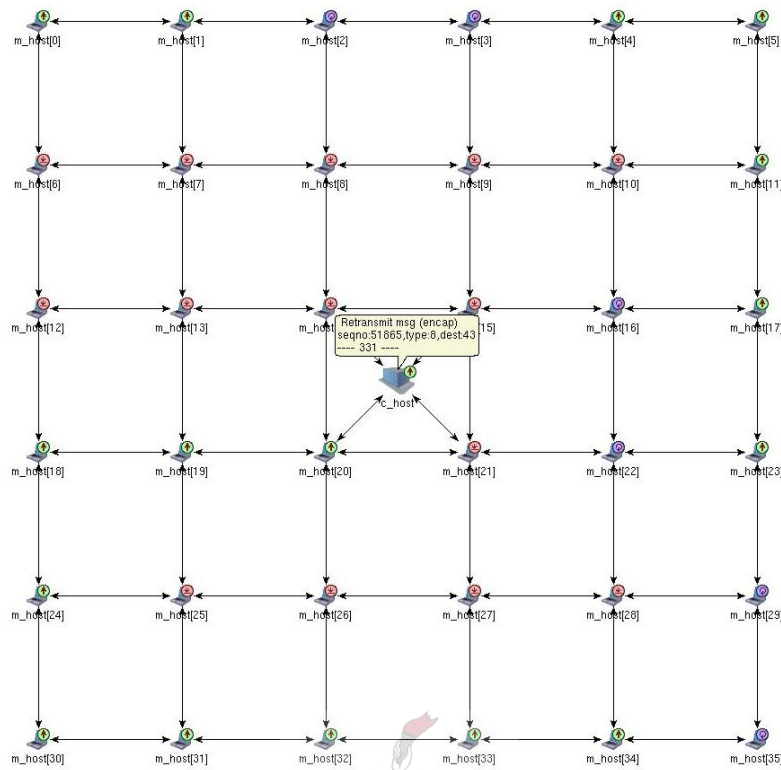


Figure 7.16: Illustration of effect of test 5 on nodes.

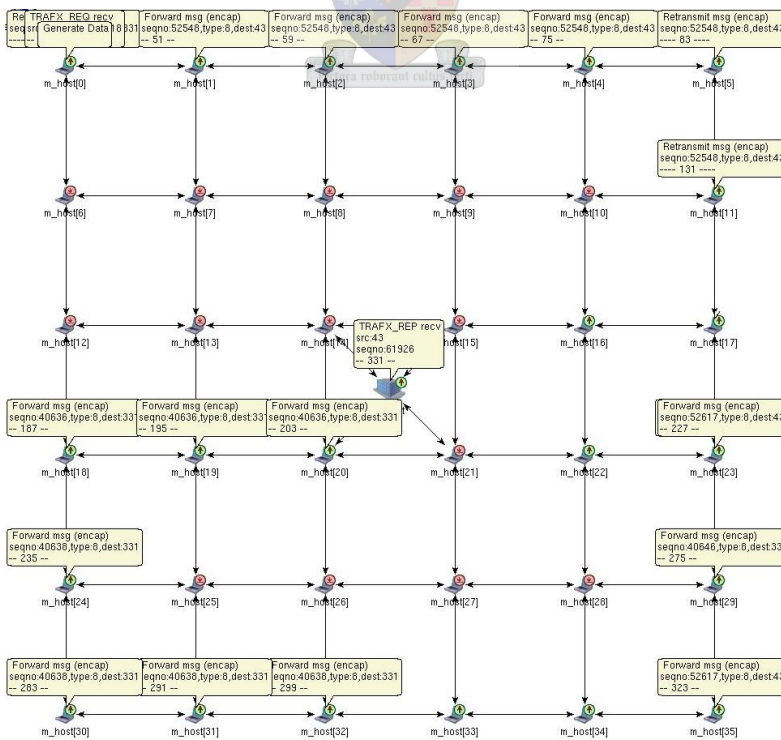


Figure 7.17: New recovered route after node failures in test 5.

7.1.8 Route metric evaluation

In the previous section, it could be seen how TH(O)RP repairs broken hierarchies and routes. In the tests conducted, TH(O)RP was forced to construct alternative routes by deactivating nodes. This however, does not conclusively verify that the route metric allows TH(O)RP to construct shortest paths on other factors than hop count alone. To verify that the metric evaluates to shortest paths based on the underlying loss link qualities, a test was conducted where a bad spot was created in the network. This was achieved, by setting the channel loss rate of three nodes on the path to node 43, to 80%. This caused a bad spot in the network around these nodes, and with the help of the route metric, TH(O)RP chose a route that avoided this area. In Figure 7.14, the original route constructed by TH(O)RP can be seen. In Figure 7.18(a), the alternative route that avoids the weak area in the network can be seen. Also, see Figure 7.18(b) for the new tree hierarchy established with the weak area included. In Figure 7.18 the weak nodes are highlighted in black circles.

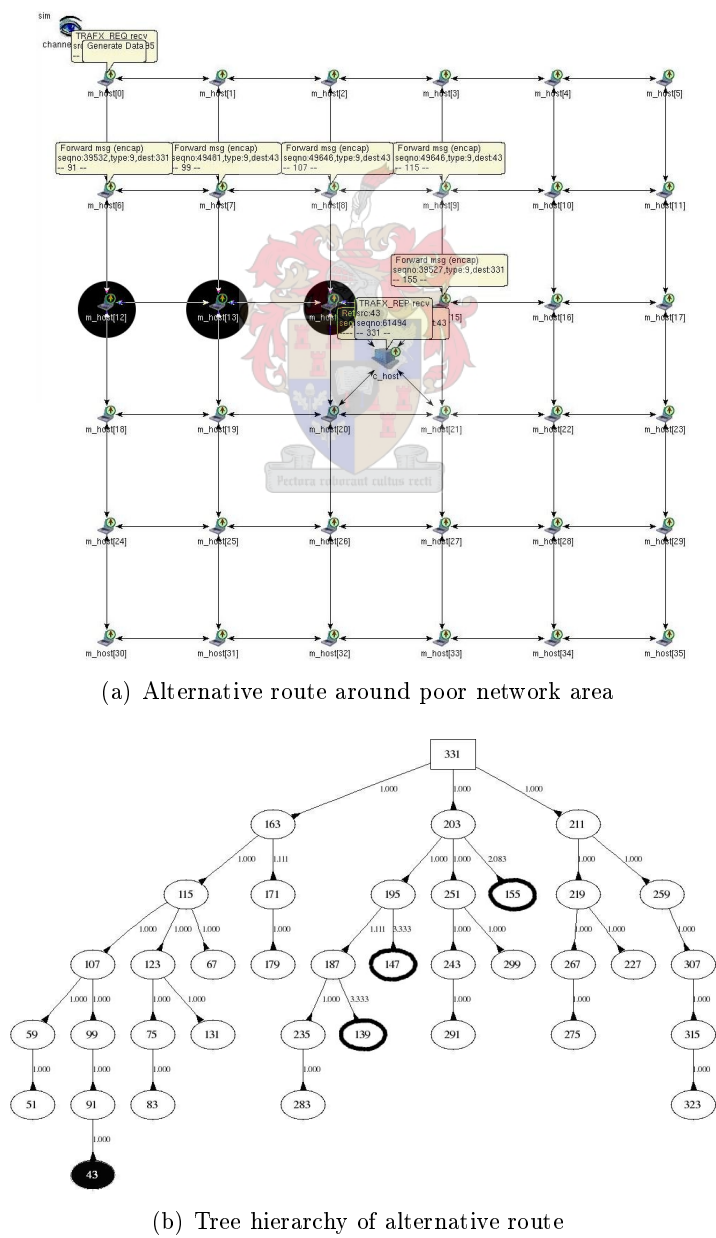


Figure 7.18: Illustration of an alternative route established with the route metric

7.1.9 Localised property of TH(O)RP

TH(O)RP is designed to localise the effects of network topology changes. This is due to the way in which the tree hierarchy is constructed, and because all messages travel up and down the hierarchy, with no horizontal message passing. It is also due to the fact that only information about the immediate neighbourhood is stored, and that a minimal amount of routing information is kept. For instance, a network change that happens on the far side of the network, with no links to the subtree containing nodes on this side of the network, will not affect the latency or throughput of these nodes. In fact, nodes on this side of the network will remain blissfully unaware of the topology changes on the far side of the network. In short, if a network topology change can be locally resolved, it will not effect the rest of the network in any way.

In order to verify this property, a test was conducted where the latency and success rate are measured for a single node, in scenario four. The test was repeated, but this time multiple nodes on the distant side of the network were deactivated after 10 minutes. The results are summarised in Table 7.11 and Figure 7.19.

The only changes that might occur, is higher up in the hierarchy and specifically at the root, which might be required to handle more or less traffic, depending on the type of modification that occurred. For instance, if nodes got deactivated, it will have less traffic, and if nodes got activated, it will have more traffic to handle. This property helps it to scale in large networks, where other protocols using full topology updates would, fail to cope with the size of the updates.

Table 7.11: *Localised property of TH(O)RP*

No disruption	Multiple node disruption
99.91%	99.91%

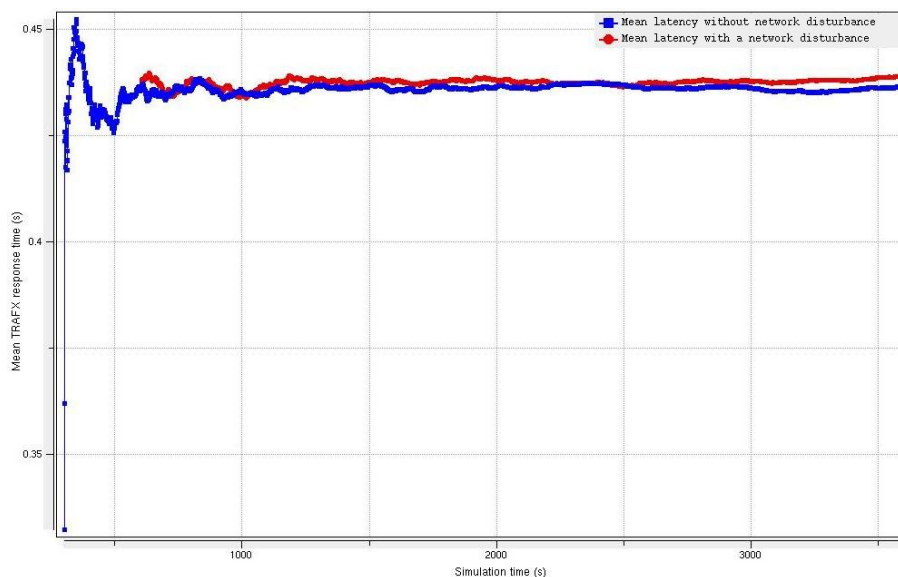


Figure 7.19: *Localised property of TH(O)RP*

7.1.10 Testing TH(O)RP scaling

The biggest limiting factors to the size of networks that ad hoc protocols can handle, is the amount of control traffic generated and the amount of information stored about the network topology. Generally, reactive routing protocols do not suffer from this scaling constraint, due to routes only being set up on demand. This is, however, a large problem in flat-routed, proactive routing protocols, that attempt to keep an up to date understanding of the entire network topology at all times. In these protocols, any topology changes are filtered through the entire network, with all nodes registering the changes. A problem is also experienced with scaling, when an attempt is made to find the shortest route, by calculating all possible routes to all destinations in the network. This causes the amount of information stored and analysed, to increase exponentially as new nodes are added to the network.

The approach used by TH(O)RP, is to only keep information about a localised area, and maintain a single optimal route to a central entity, whilst maintaining suboptimal routes to all other network nodes. The amount of information stored per node, is also varied. For instance, nodes located high up in the tree hierarchy will store more routes, namely all the routes of nodes below it in its branch of the tree, than nodes found at the bottom of the hierarchy. Due to this, there is a linear increase in the amount of information stored, as the addition of a single node only requires a single route entry to be added to nodes located higher up in the tree hierarchy. This node would not be registered in the tables of nodes lower in the hierarchy, unless the new node is located in the neighbourhood of the lower nodes. A single new node, only introduces a single increase in the number of route request messages in the network. The same holds for adding N nodes to the network.

The scalability of TH(O)RP can be visually verified, by looking at the network layer output during the final simulation scenario. This scenario has 100 nodes placed randomly around the network. See Figure 7.20 for the network layer output of the instation, an outstation high up in the hierarchy and an outstation at the bottom of the hierarchy. The simulation was conducted, with polylogue messages being transmitted to the outstation at the bottom of the hierarchy. Inspection of the amount of control traffic generated and processed by these nodes, points out the difference in traffic at different levels of the hierarchy. In Table 7.12, it can be seen that the instation handles a considerable amount more control traffic than the other two nodes, with the outstation at the bottom of the hierarchy handling the least amount of control traffic. In Table 7.13, the route table storage requirement was calculated for the three nodes. This table also indicates, that route storage requirements decrease dramatically when moving down the tree hierarchy. As mentioned earlier, TH(O)RP can theoretically scale to large networks, which is a necessary requirement for the proposed implementation, where hundreds of TICs could exist in the network. Once again, this could not be verified, due to processing and memory restrictions on the test machine.

```

(THRPNetvLayer) sim_c_host.net (id=907) (pdr0x8790c80)

--- 98.055s ---ROOT-----RETRY TABLE
Destination IP Sequence number Msg type Retries Time out
--- 98.055s ---ROOT-----
--- 98.055s ---ROOT----- DUP TABLE
Originator IP Sequence number Time out
--- 98.055s ---ROOT-----
--- 98.055s ---ROOT----- LINKS
IP address hyst LQ lost total NLO ETX
491 0.353 0.300 1 10 1,000 1,111
387 1,000 1,000 0 10 1,000 1,000
283 0.393 0.300 1 10 1,000 1,111
531 0.377 0.800 2 10 0,700 1,785
--- 98.055s ---ROOT-----
--- 98.055s ---ROOT----- NEIGHBORS
IP address CHLD PRNT SYN HPR NCHLD NRCOST TRCOST
387 YES NO YES YES 0 2,500 inf
491 YES NO YES YES 0 2,000 inf
531 YES NO YES YES 7 2,000 inf
283 YES NO YES YES 1 3,125 inf
--- 98.055s ---ROOT-----
--- 98.055s ---ROOT----- ROUTES
NH IP addr Dest IP addr HOPCNT INTF WCETX
531 443 2 eth0 2,000
531 157 2 eth0 2,000
531 203 2 eth0 2,000
531 139 2 eth0 2,111
531 893 2 eth0 2,000
531 155 2 eth0 2,000
531 511 3 eth0 3,786
531 467 3 eth0 3,250
531 235 3 eth0 3,500
531 639 3 eth0 3,786
531 983 3 eth0 3,222
531 371 3 eth0 3,235
531 323 3 eth0 4,340
531 243 3 eth0 3,250
531 723 3 eth0 3,786
531 773 2 eth0 2,000
531 597 3 eth0 3,250
531 147 3 eth0 3,250
531 843 3 eth0 3,250
531 815 4 eth0 4,235
531 627 4 eth0 4,222
531 153 4 eth0 5,035
531 899 4 eth0 4,472
531 579 4 eth0 4,651
531 293 5 eth0 5,596
531 615 5 eth0 5,722
531 827 5 eth0 5,722
531 539 4 eth0 5,035
531 631 5 eth0 6,303
531 739 5 eth0 6,734
531 811 4 eth0 4,596
531 403 5 eth0 5,885
531 667 6 eth0 6,685
531 339 6 eth0 6,818
531 287 6 eth0 6,818
531 795 5 eth0 5,750
531 211 5 eth0 6,040
531 475 6 eth0 7,274
531 363 6 eth0 6,985
531 251 6 eth0 6,830
531 683 6 eth0 7,361
531 583 6 eth0 6,981
531 523 6 eth0 6,830
531 891 6 eth0 7,207
531 715 6 eth0 7,984
531 427 7 eth0 8,231
531 747 6 eth0 5,707
531 463 6 eth0 7,250
531 275 7 eth0 7,611
531 307 7 eth0 9,115
531 975 7 eth0 9,775
531 803 7 eth0 8,540
531 787 7 eth0 7,611
531 179 6 eth0 7,423
531 851 6 eth0 7,423
531 659 6 eth0 7,423
531 935 7 eth0 9,863
531 171 7 eth0 8,429
531 515 7 eth0 8,540
531 107 7 eth0 9,429
531 795 6 eth0 7,429
531 131 7 eth0 8,429
531 936 7 eth0 9,429
531 771 5 eth0 6,286
531 227 7 eth0 8,540
531 315 6 eth0 7,207
531 115 7 eth0 8,679
531 763 5 eth0 6,734
531 651 7 eth0 9,313
531 299 7 eth0 8,194
283 331 2 eth0 3,125
283 459 3 eth0 4,125
283 507 3 eth0 3,639
283 123 4 eth0 5,236
283 411 3 eth0 4,125
    
```

(a) Instation output

```

(THRPNetvLayer) sim_m_host[75].net (id=707) (pdr0x872fa48)

--- 98.075s ---ROOT-----RETRY TABLE
Destination IP Sequence number Msg type Retries Time out
--- 98.075s ---ROOT-----
--- 98.075s ---ROOT----- DUP TABLE
Originator IP Sequence number Time out
619 44163 106,2961
619 44156 100,0987
315 59297 106,5424
315 59279 100,6537
563 8959 100,1273
563 8956 106,2077
251 50295 106,1532
827 65295 100,1273
251 50298 100,1273
827 65302 106,2517
907 12879 100,7037
907 12869 100,7037
907 12856 100,8182
907 12836 100,8182
907 12837 100,8182
907 12829 100,8182
907 12304 100,8182
907 12345 100,8182
907 12310 100,8337
259 34474 106,1132
259 34467 100,3559
931 13775 100,1273
931 13782 106,5424
923 54235 100,1273
923 54242 106,1532
683 7086 100,1273
683 7093 106,4474
--- 98.075s ---ROOT-----
--- 98.075s ---ROOT----- LINKS
IP address hyst LQ lost total NLO ETX
619 0.750 0.300 1 10 1,000 1,111
587 0.449 0.700 3 10 1,000 1,429
547 0.299 1,000 0 10 1,000 1,000
499 0.388 0.800 2 10 1,000 1,250
387 0.397 0.300 1 10 0,300 1,235
259 0.623 0.800 2 10 0,800 1,562
547 0.390 0.300 1 10 1,000 1,111
827 0.682 0.600 4 10 0,800 2,083
--- 98.075s ---ROOT-----
--- 98.075s ---ROOT----- NEIGHBORS
IP address CHLD PRNT SYN HPR NCHLD NRCOST TRCOST
619 NO NO YES YES 0 12,569 13,819
259 YES NO YES YES 0 12,816 14,066
547 NO NO YES YES 0 10,247 11,497
619 YES NO YES YES 0 12,160 13,410
587 NO NO YES YES 0 7,844 8,894
499 NO YES YES YES 4 6,336 9,189
347 NO NO YES YES 0 10,247 11,497
827 YES NO YES YES 6 12,160 13,410
--- 98.075s ---ROOT-----
--- 98.075s ---ROOT----- ROUTES
NH IP addr Dest IP addr HOPCNT INTF WCETX
499 307 4 eth0 5,575
827 251 2 eth0 2,663
827 803 2 eth0 3,096
827 963 2 eth0 2,817
827 823 2 eth0 2,663
827 881 2 eth0 2,301
827 315 2 eth0 2,301
    
```

(b) Middle outstation output

```

(THRPNetvLayer) sim_m_host[87].net (id=803) (pdr0x875e768)

--- 98.052s ---ROOT-----RETRY TABLE
Destination IP Sequence number Msg type Retries Time out
--- 98.052s ---ROOT-----
--- 98.052s ---ROOT----- DUP TABLE
Originator IP Sequence number Time out
--- 98.052s ---ROOT-----
--- 98.052s ---ROOT----- LINKS
IP address hyst LQ lost total NLO ETX
651 1,000 1,000 0 10 0,300 1,111
475 1,000 1,000 0 10 0,800 1,250
835 0.364 0.300 1 10 0,300 1,235
--- 98.052s ---ROOT-----
--- 98.052s ---ROOT----- NEIGHBORS
IP address CHLD PRNT SYN HPR NCHLD NRCOST TRCOST
835 NO NO YES YES 0 16,770 18,020
651 NO NO YES YES 0 16,770 18,020
475 NO YES YES YES 3 13,667 14,917
--- 98.052s ---ROOT-----
--- 98.052s ---ROOT----- ROUTES
NH IP addr Dest IP addr HOPCNT INTF WCETX
475 307 7 eth0 8,540
    
```

(c) Bottom outstation output

Figure 7.20: Network layer output of nodes at different levels of tree hierarchy

Table 7.12: Control traffic rates at different levels in tree hierarchy

Node hierarchy level	Packets Th(o)rp	Used bandwidth Control	Used bandwidth Data	Data to control Ratio
Top (root)	6400	0.4132	0.3386	0.8194
Middle	5705	0.2759	0.6641	2.4072
Lowest	3092	0.0833	0.3303	3.9677

Table 7.13: Route table storage requirements

Node hierarchy level	Number of routes (N)	Number of destinations (M)	RT_{size} (bits)	RT_{size} (bytes)
Top (root)	4	96	66144	8268
Middle	2	7	3680	460
Lowest	1	1	1344	168

7.1.11 Effect of startup time

In most network environments, nodes are generally switched on at different times. This is, however, not the case in simulations, where all nodes are activated at the exact same time. Because all nodes in the network have equal message- and event generation intervals, similar messages and events tend to coincide at nodes across the network. Although the unintentional synchronised message and event generation intervals should not affect the performance of TH(O)RP, the effect of desynchronising these events was still tested.

The test was conducted on scenario four, with all nodes supplied with a random startup time, between 1 and 100 seconds. This was compared a test with all nodes active at the start of the simulation. The results are summarised in Table 7.14. From the results, it can be noted that there is merely a negligible difference, with TH(O)RP still operating smoothly, even though events are unsynchronised.

Table 7.14: Synchronised versus unsynchronised startup times for nodes

Startup	Success rate
Synchronised	88.239%
Unsynchronised	88.523%

7.1.12 Summary

The initial simulations, optimised the parameters of the TH(O)RP protocol. It also pointed out the limitations, on the number of network hops over which a 1 second maximum latency can be guaranteed. While most of the simulation results were favourable, it was discovered that TH(O)RP is sluggish to respond to network changes, due to its periodic nature. In the following section, methods are developed to improve this response.

7.2 Introduction of the fast recovery methods

As mentioned above, the initial simulations indicated that TH(O)RP was sluggish to respond to network topology changes. In view of this poor response, an investigation into additional methods and optimisations was conducted. In this section the methods are discussed, and in the next section the impact of these methods are evaluated.

7.2.1 Overview

This section, develops methods to improve the responsiveness of TH(O)RP to network topology changes. These methods include, sending out messages informing nodes that a route error has occurred, temporarily increasing the periodic updates, including additional information in existing messages and sending special notification messages. In all, five methods were developed. These methods were not developed to address the issue of speeding up the discovery of a route failure. They were developed to increase the speed at which the protocol can recover once an error is discovered. All of these methods were dubbed fast recovery methods.

As will be seen in Section 7.3, the fast recovery methods increase the response of the protocol, as well as improving the overall success rate and latency of the network, because TH(O)RP was better equipped to react to network topology changes.

7.2.2 Method 1: Variable parent selection interval

The first method to increase responsiveness, was to introduce a variable parent selection interval. With the original, static update interval of 6 seconds, a node could be required to wait up to 6 seconds when losing a parent, before regaining connectivity to the network. The static update interval, also had a detrimental effect on the construction of the original hierarchy, as the hierarchy is passively built from top to bottom. It could therefore take up to 6 seconds per network hop, to construct the full tree hierarchy.

It was decided, to let each node be initialised with a shortened parent selection interval of 2 seconds, which is the same as the HELLO generation interval. This would provide faster reactions, to nodes in a neighbourhood that become eligible parents. Also, when a node loses a parent and is unable immediately to select a new parent, it will again use a shortened parent selection interval. In short, when a node does not have a valid parent, it will use a shortened parent selection interval.

7.2.3 Method 2: Assisted high speed propagation path (AHSP)

During simulation it was noted, that when a node selects a parent, there was a delay in the realisation by the parent of the new child. This was due to the fact, that no active mechanism exists to notify a parent when it is selected as a parent. The notification is passively accomplished, through the periodic HELLO messages. This had the adverse effect, that the initial route request issued by a node when selecting a new parent, was ignored by that parent because it has not yet registered the node as a child.

It was decided to add additional information to route request messages, informing a newly selected parent that it was selected as a parent. This removed the wait-time for the next HELLO message to be generated, and had the effect that a parent immediately became aware of a new

child, as soon as that child selects it as parent and issues a route request message. AHSP also had the effect that the network hierarchy became more rapidly constructed, with the instation becoming aware of new visible nodes faster.

7.2.4 Method 3: Adoption notification

When a node selects a parent node, it becomes eligible for selection as a parent itself. However, this was only made known the next time the node generated a HELLO message. It was decided to issue a special HELLO message, as soon as a node selects a parent, announcing that it can now be selected by other nodes as their parent. This drastically increased the rate at which the network topology could be constructed.

In addition to announcing when a node selects a new parent, the instation was modified to issue a HELLO message as soon as it became active, reducing the amount of time it took for the surrounding outstations to become aware of the instation, and of the fact that it can be selected as a parent.

7.2.5 Method 4: Route error notification

When a node high up in the tree hierarchy loses its parent and is unable to select a new parent, it becomes an orphan. The nodes lower down in the hierarchy, would only realise that their route is no longer valid, when the newly orphaned node sends out its next HELLO message. This had the effect that a subtree would gradually deteriorate from top to bottom, until a node lower down the subtree is able to select a new parent. This caused a long delay in new routes being acquired.

When a node is orphaned, it is still active and can still send out messages. It was decided to introduce a route error notification message, RERR, that would inform neighbouring nodes and nodes lower down the hierarchy, that a node has lost its parent and cannot select a new parent. When a node receives a route error message, it tries to select a parent and if it is able to do so, will send a route fixed message back to the issuer of the original route error message. If it is not able to select a valid parent, it sends the route error message further down the hierarchy. This reduced the time it took for nodes lower down the hierarchy to discover that their routes had failed, and allowed them to set up alternative route faster. If none of the nodes are able to select a new parent, they will all become orphans until one can select a new parent.

Introducing route error notifications, had the beneficial effect of removing the issue of nodes selecting other nodes that have lost their parent, and have not realised it yet, which allowed the recovery process to start more promptly.

7.2.6 Method 5: Cumulative route request (CRREQ)

When a node high up in the tree hierarchy selects a new parent, it pushes a route for itself up the tree hierarchy. Due to the fact that it was able to immediately select a new parent, all the nodes beneath this node maintain their original routes and still view them as valid. This is true for a view from the bottom to the top of the hierarchy, but the routes stored at the root of the hierarchy will now be old and invalid.

To remedy this issue, a new message type was declared, that is issued when a node selects a new parent and has nodes beneath it in the hierarchy, or more specifically when it has children. In

the event of this occurring, the node will issue a cumulative route request (CRREQ), containing updates for all the route entries of this node. When nodes higher up the hierarchy receives this message, they can update their routing tables for the node that selected a new parent and all of the nodes beneath it. This provides the routes and hierarchy, with added robustness and adaptability to changing topologies.

7.2.7 Summary

All of these methods, had beneficial effects on the overall speed at which TH(O)RP can construct its hierarchy and recover from network failures. The first three methods are used to facilitate faster hierarchy construction, and the fourth method reduces the amount of time required to deteriorate and recover subtrees. The final method aims to improve network stability, by updating multiple routes, to reflect the current network topology. In the original form of TH(O)RP, the hierarchy in scenario four took approximately 36 seconds to be constructed. After the introduction of these methods, this time was reduced to 10 seconds. It can be argued, that with further fine tuning of timing parameters and event synchronisation, this time could be decreased even further. This was however, not pursued any further. It should be mentioned, that the adoption announcement method relies on the successful delivery of HELLO messages, which are transmitted only once, in order to be effective. This could reduce its effectiveness in lossy networks.

7.3 Revisited simulation results

7.3.1 Overview

In Section 7.1.7, responsiveness tests were conducted. It was found, that TH(O)RP was sluggish to respond to network topology changes due to its periodic nature. In the previous section, five methods were developed to speed up its response.

In this section, the impact of these methods on the overall response and performance of TH(O)RP, are measured. Firstly, the effect of the methods to the responsiveness of TH(O)RP, is measured. Secondly, a test is conducted to verify that the methods do not negatively effect the overall performance of TH(O)RP. These tests were conducted with scenarios four, and two through five, respectively.

7.3.2 Responsiveness tests revisited

This test, compares the effect of individual or all fast recovery methods, to results where fast recovery methods are not used. The results are summarised in Table 7.15. It should be noted that the adoption announcement method was unable to recover from the final test when used alone, resulting in the broken hierarchy never being reconstructed. The route error notification method, also introduced a longer latency when used on its own.

From these tests it can be seen, that the improvement is related to the severity of the disruption, resulting in the best improvement observed when the network topology is almost completely broken. This is due to the fact that the hierarchy had to be reconstructed from scratch. Some methods are more successful than others, and some are not usable by themselves.

Table 7.15: *Route recovery times revisited*

Test	None	Increased parent selection interval	AHSP	Adoption announcement	RERR notification	CRREQ	All
1	18s	18s	16s	14s	14s	14s	14s
2	18s	18s	18s	18s	18s	12s	14s
3	18s	14s	14s	14s	14s	12s	14s
4	18s	14s	14s	14s	14s	12s	14s
5	126s	40s	79s	Could not recover	146s	108s	24s

This once again, pointed out the fact that these methods speed up recovery from registered errors, and does not speed up error detection.

The amount of time required to recover from a disruption, also depends on when errors occur between update intervals. If errors occur directly after the last update, a node will wait the full update interval before reanalysing the network. If it happens just before an update, a node will have a much shorter period before reanalysing the network, and will discover route errors faster.

This once again, points out the importance of speeding up the discovery of network topology changes and disruptions. This will prove difficult to solve, because if a route is considered lost every time a message is lost, a situation will develop where the network topology constantly oscillates, which will negatively effect network stability. A method specifically designed to prevent this from happening, was implemented in the form of link hysteresis, which delays link state changes. Unfortunately, this had the detrimental side-effect of making TH(O)RP sluggish to respond to real network failures. Seemingly, increasing the route error detection rate will decrease the network stability, which is undesirable.

A better solution would be to introduce route redundancy, keeping multiple paths, or multiple tree hierarchies, to provide a quick alternative route in the event of network failures.

7.3.3 Overall impact of fast recovery on TH(O)RP performance

This test, measures the effect that the fast recovery methods on the performance of TH(O)RP. As was shown in the previous test, the best results were obtained when all the fast recovery methods were used together. Thus, the test is only conducted for all the fast recovery methods activated. This test uses the default timing parameters, as derived through optimisations in Section 7.1. It compares the performance of the protocol, with no fast recovery methods enabled, versus all fast recovery methods enabled, and with a channel drop-rate of 10%. This tests was conducted for all the simulation scenarios. The results are summarised in Table 7.16. Because the network was not disrupted during any of these tests, the resulting latencies do not show any noticeable differences and were not included.

Although the improvements of the fast recovery are not drastic, they do bring a small improvement to the success rate of TH(O)RP. This is due to the fact, that TH(O)RP is able to faster construct the network topology and can mimic the underlying network topology more efficiently.

Table 7.16: *Effect of fast recovery methods on TH(O)RP*

Test scenario	No fast recovery methods enabled	All fast recovery methods enabled
1	95.13672%	95.32081%
2	93.76351%	93.96006%
3	60.99435%	61.76109%
4	88.94523%	89.19999%
5	76.78612%	76.88580%

7.3.4 Summary

In this section, the effects of the fast recovery methods on the responsiveness of TH(O)RP, was illustrated. Also, it was found that this time could not be reduced beyond a certain minimum, without effecting the network stability. The results is, that TH(O)RP will not be able to guarantee a 1 second maximum latency under all operating conditions. This can however, be ensured under normal operating conditions, while valid routes exist. Expecting the protocol to recover from various types of network disruptions, in under 1 second under all conditions is unrealistic, and would severely handicap the performance of the protocol, as it would become unstable. One method that could reduce the effect of network disruptions would be to introduce route redundancy, which would also make the protocol more robust.

In the end, it was decided to promote a stabilised network hierarchy over the speed of responsiveness to network disruptions. This is due to the fact that the network topology can be designed to minimise the probability of network disruptions, through the correct use of node and antenna placement.

7.4 Conclusion

In this chapter, the TH(O)RP protocol was simulated, optimising the initial timing parameters, measuring the protocols reactions and the effect of multiple hops. The scaling attribute of TH(O)RP was also verified. Some deficiencies, in the reaction time of TH(O)RP became apparent during these tests, and methods were developed to address these issues.

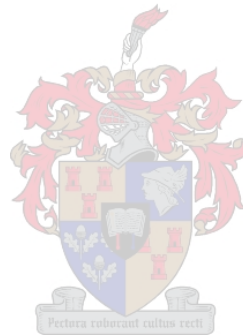
One of the main issues that have yet to be addressed is the timely discovery of network topology changes, but this, however, proves difficult in a protocol that has a periodic nature without resorting to overloading the network with unnecessary control traffic. It was proposed, that route redundancy should rather be investigated as a solution to this problem, as it would increase the protocol robustness and could provide quick, alternative routes to be used in the event of the main route failing.

TH(O)RP was able to deliver on the low latency requirement, with RTTs lower than 1 second up to 11 hops under normal operating conditions, and up to 8 hops when links are subjected to a 10% drop rate. Higher latencies were experienced when route failures occurred. The probability of node and link failures can be minimised, by ensuring that the network has good wireless links, through good antenna placement. But eventually, the protocol will be required to recover from a network failure of some sort. The effect of link failures can be limited by introducing route redundancy, as mentioned earlier, and this would be part future research and improvements planned for TH(O)RP.

The protocol did scale gracefully to large networks without an overall performance loss, and this proved the applicability of TH(O)RP to large networking environments, as would be required to provide the communication backbone for the TICs of a metropolitan area.

Another point that the tests illustrated, was the localised manner in which TH(O)RP operates and that changes to the network topology unrelated to other areas of the network, do not effect those areas in any way at all. This also contributes to the scalability of TH(O)RP.

Overall, TH(O)RP has proven that it is able to control the communication backbone for the desired application, and with a few improvements it can become an efficient solution to address the issues mentioned in this thesis.



Chapter 8

C implementation of TH(O)RP

8.1 Introduction

A C implementation of TH(O)RP was based on the open source project found in [46]. Implementing the protocol in C provides a lightweight implementation with few dependencies, with the result that it should be compatible with various hardware platforms and embedded systems. The current version only runs on the Linux operating system.

The C implementation of TH(O)RP, is currently not as complete as the C++ implementation developed for the simulations. Some of the missing features include the delivery system, the retransmissions and acknowledgements, and the fast recovery methods developed during the simulations. A TRAFX compatability layer has yet to be developed for the C implementation of TH(O)RP. The plug-in interface has also not been fully implemented and is currently not operational.

In this chapter, the development of the C implementation of TH(O)RP is briefly discussed, starting with the composition of the implementation. The output information available during the execution of TH(O)RP, is also discussed. The different user input parameters available for TH(O)RP, are discussed. A virtual network emulator, the `thrp_switch`, which is based on a network emulator found in [46] is in Section 8.5. Finally, future developments planned for TH(O)RP are discussed.

8.2 Application composition

The C implementation of TH(O)RP follows the same modular design, as described in Section 4.8. It comprises the socket parser, packet parser, information repositories and scheduler explained in Section 4.8. Various other modules used to generate messages, communicate with networking hardware, set defaults for the configuration and display debugging information are also implemented.

8.3 Application output

The C implementation of TH(O)RP can generate various levels of output during operation. The degree of output information that TH(O)RP generates, is controlled by input parameters that can be set by the user. See Section 8.4 for a description of the different user inputs available.

To illustrate the output generated by TH(O)RP, a small example network was created with six nodes which communicated over the `thrp_switch` network emulator. The target tree hierarchy is displayed in Figure 8.1.

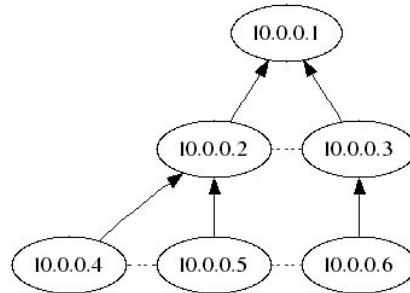


Figure 8.1: Target network hierarchy for illustration of TH(O)RP output

8.3.1 Information repository output

The table outputs of the C implementation of TH(O)RP, are similar to the output of the C++ implementation used during simulations. Differences are that the C++ implementation had outputs for its information repositories, which included the neighbour-, route and retry table, and the link- and duplicate set. The C implementation has fewer information repositories at present, including the neighbour- and route tables, and the link- and duplicate set. Additionally it keeps a mid set. Output during execution of the network of Figure 8.1 for nodes 10.0.0.1, 10.0.0.2 and 10.0.0.4, are displayed in Figure 8.2.

From the output it can be seen that node 10.0.0.1, defined as the root, has route entries to nodes 10.0.0.4, 10.0.0.5 and 10.0.0.6 as expected. Node 10.0.0.4 has a route to node 10.0.0.1 at the top of the hierarchy.

8.3.2 TH(O)RP packet output

During the execution of TH(O)RP, output for messages sent and received can be enabled. This is achieved by enabling the `-dispout` and `-dispin` switches respectively when executing TH(O)RP. See Section 8.4 for the complete list of switches for TH(O)RP.

Printing out the contents of sent and received messages, aids in the debugging of TH(O)RP operation. In Figure 8.3 three TH(O)RP packets can be seen, each containing a HELLO, RREQ or RREP message respectively.

```

*** THRP - 0.1.00 (Nov 3 2006) ***

17:51:17.54 ---ROOT--- LINKS
IP address      hyst  LQ    lost  total  NLQ    ETX
10.0.0.3        1.000  1.000  0     10     0.890  1.114
10.0.0.2        0.980  0.900  1     10     0.890  1.237

17:51:17.54 ---ROOT--- NEIGHBORS
IP address      CHLD  PRNT  SYN   HPR   NCHLD  NRCOST  TRCOST
10.0.0.2        YES  NO    YES  YES  2      1.111  2.348
10.0.0.3        YES  NO    YES  YES  1      1.237  2.351

17:51:17.54 ---ROOT--- ROUTES
NH IP addr      Dest IP addr  HOPCNT  INTF   WCETX
10.0.0.2        10.0.0.4     2       hciF01 2.000
10.0.0.2        10.0.0.5     2       hciF01 2.000
10.0.0.3        10.0.0.6     2       hciF01 2.000
    
```

(a) Output of node 10.0.0.1

```

*** THRP - 0.1.00 (Nov 3 2006) ***

17:50:47.86 ---ROOT--- LINKS
IP address      hyst  LQ    lost  total  NLQ    ETX
10.0.0.5        0.867  0.800  2     10     0.800  1.562
10.0.0.4        1.000  1.000  0     10     0.890  1.114
10.0.0.3        1.000  1.000  0     10     0.498  2.000
10.0.0.1        0.990  0.800  2     10     0.800  1.562

17:50:47.86 ---ROOT--- NEIGHBORS
IP address      CHLD  PRNT  SYN   HPR   NCHLD  NRCOST  TRCOST
10.0.0.1        NO    YES  YES  YES  2      0.000  1.562
10.0.0.3        NO    NO   YES  YES  1      1.389  3.397
10.0.0.4        YES  NO   YES  YES  0      2.111  3.225
10.0.0.5        YES  NO   YES  YES  0      2.392  3.954

17:50:47.86 ---ROOT--- ROUTES
NH IP addr      Dest IP addr  HOPCNT  INTF   WCETX
10.0.0.1        10.0.0.1     0       hciF01 0.000
10.0.0.1        10.0.0.2     2       hciF01 2.000
10.0.0.1        10.0.0.3     2       hciF01 2.000
10.0.0.1        10.0.0.4     2       hciF01 2.000
10.0.0.1        10.0.0.5     2       hciF01 2.000
    
```

(b) Output of node 10.0.0.2

```

*** THRP - 0.1.00 (Nov 3 2006) ***

17:48:55.00 ---ROOT--- LINKS
IP address      hyst  LQ    lost  total  NLQ    ETX
10.0.0.5        0.625  0.800  2     10     0.596  2.097
10.0.0.2        0.746  0.800  2     10     1.000  1.250

17:48:55.00 ---ROOT--- NEIGHBORS
IP address      CHLD  PRNT  SYN   HPR   NCHLD  NRCOST  TRCOST
10.0.0.2        NO    YES  YES  YES  2      1.237  2.487
10.0.0.5        NO    NO   NO   YES  0      2.237  4.334

17:48:55.00 ---ROOT--- ROUTES
NH IP addr      Dest IP addr  HOPCNT  INTF   WCETX
10.0.0.2        10.0.0.1     2       hciF01 2.000
    
```

(c) Output of node 10.0.0.4

Figure 8.2: Network layer output during execution of TH(O)RP

```

===== THRP PACKET =====
source: 10.0.0.1
length: 76 bytes
seqno: 13

----- THRP MESSAGE -----
Sender main addr: 10.0.0.10
Type: LQ-HELLO, size: 72, utime: 6.00
TTL: 1, Hopcnt: 0, seqno: 28965
+HTime : 5.000
+Will : 5.000
+HParent : YES
+Children: 1
+RCost : 0.000
+Parent : 10.0.0.10
    
```

(a) HELLO message

```

===== THRP PACKET =====
source: 10.0.0.5
length: 36 bytes
seqno: 122

----- THRP MESSAGE -----
Sender main addr: 10.0.0.5
Type: RREQ, size: 32, utime: 0.06
TTL: 255, Hopcnt: 0, seqno: 57554
+WCETX : 2.000
+Next hop : 10.0.0.10
+Source : 10.0.0.5
+Destination: 10.0.0.5
    
```

(b) RREQ message

```

===== THRP PACKET =====
source: 10.0.0.1
length: 36 bytes
seqno: 14

----- THRP MESSAGE -----
Sender main addr: 10.0.0.10
Type: RREP, size: 32, utime: 0.06
TTL: 255, Hopcnt: 0, seqno: 28966
+WCETX : 2.000
+Next hop : 10.0.0.6
+Source : 10.0.0.10
+Destination: 10.0.0.5
    
```

(c) RREP message

Figure 8.3: TH(O)RP packets and messages

8.4 Execution and input options

The C implementation of TH(O)RP is executed from the command line, under a Linux operating system. Due to socket bindings made for communicating with the network interfaces, it must be executed by a root- or super user. This can be accomplished, by switching to a root user using either ‘sudo’ or ‘su’, and then executing TH(O)RP with the desired input parameters. For example, to run TH(O)RP with an Ethernet interface and with moderate debugging information output and with the routing metric enabled, the following is entered:

```
sudo ./thrp -i eth0 -d 3 -L 1 -W 10
```

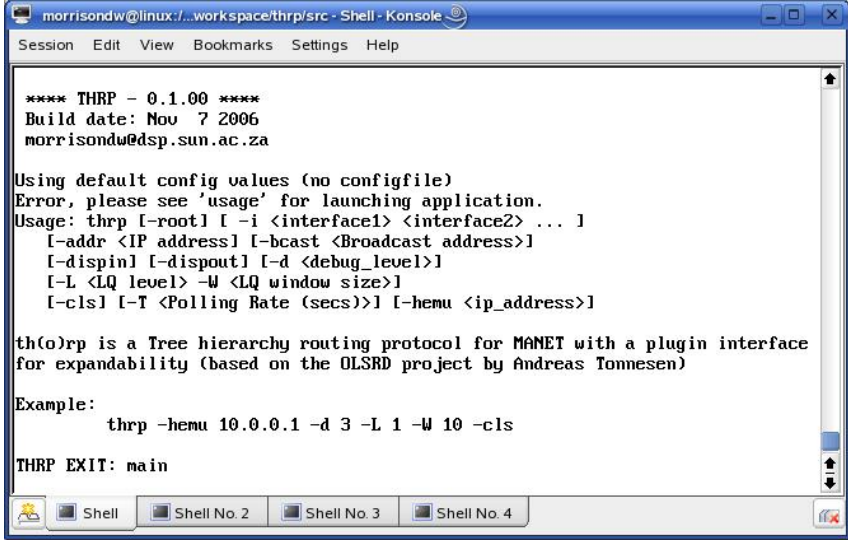
where ‘eth0’ is the name of the Ethernet interface under a Linux operating system. Interface names can be displayed by executing the ‘ifconfig’ command as a super user. Various settings of TH(O)RP can be set from the command line. See Figure 8.4 for the usage summary of TH(O)RP. These input parameters are used to override some of the default settings of TH(O)RP maintained internally. More user defined override values and possibly a configuration file reader is planned for future versions of TH(O)RP. The different input parameters are briefly described in Table 8.1.

Table 8.1: *Command line input parameters for TH(O)RP*

Input	Description
-root	Set this node as the ROOT of the tree hierarchy
-i	Define network interface(s) to use with TH(O)RP, ex. ‘-i eth0’
-addr	Manually define the IP address of this node
-bcast	Manually define the broadcast address of this node
-dispin	Display incoming TH(O)RP packets
-dispout	Display outgoing TH(O)RP packets
-d	Set the debugging level (0 for no output)
-L	Set to 1 to use link quality routing (ETX and WCETX metric)
-W	Set the window size for link quality routing
-cls	Enable clear screen output, for better output of information repositories
-T	Set the polling interval of the scheduler
-hemu	Set a software emulated network interface and IP address (used in conjunction with thrp_switch)

8.5 Virtual network: thrp_switch

The thrp_switch, is based on a network emulator from [46]. It simulates a network environment through which multiple TH(O)RP nodes can communicate. A hardware emulated interface can be defined as the network interface of each TH(O)RP node, by using the ‘-hemu’ input parameter and specifying an IP address when TH(O)RP is executed. This creates a network socket that connects to the thrp_switch server, and the TH(O)RP protocol then communicates with other connected nodes through the switch. Employing the simulated network environment, multiple



```

morrisondw@linux:~/workspace/thrp/src - Shell - Konsole
Session Edit View Bookmarks Settings Help

**** THRP - 0.1.00 ****
Build date: Nov 7 2006
morrisondw@dsp.sun.ac.za

Using default config values (no configfile)
Error, please see 'usage' for launching application.
Usage: thrp [-root] [-i <interface1> <interface2> ... ]
      [-addr <IP address>] [-bcast <Broadcast address>]
      [-dispin] [-dispout] [-d <debug_level>]
      [-L <LQ level>] [-W <LQ window size>]
      [-cls] [-T <Polling Rate (secs)>] [-hemu <ip_address>]

th(o)rp is a Tree hierarchy routing protocol for MANET with a plugin interface
for expandability (based on the OLSRD project by Andreas Tonnesen)

Example:
      thrp -hemu 10.0.0.1 -d 3 -L 1 -W 10 -cls

THRP EXIT: main

```

Figure 8.4: Usage summary of TH(O)RP

TH(O)RP nodes can be executed on a single computer, which helps in the development of the protocol.

The `thrp_switch`, also provides various commands for controlling the nature of the communication between nodes connected to the switch. In Figure 8.5, the commands used to achieve the network environment that produced the hierarchy of Figure 8.1, can be seen. The first command, sets the link qualities between all nodes in both directions to drop 10 out of every 100 packets, which creates links with a 10% drop rate. The next command sets the link quality between node 10.0.0.1 and 10.0.0.4 to 0, which closes the link. The remaining commands set the remaining links to achieve the desired network topology. It should be noted that the `thrp_switch` does not create the network hierarchy; it merely sets the network topology. TH(O)RP then creates the hierarchy from the network topology.

Other commands for the `thrp_switch` are `help`, `exit`, `log`, `list`, and `link`. The documentation for these commands can be found in [47], which is the documentation for the program upon which the `thrp_switch` is based.

```

link bi * * 10

link bi 10.0.0.1 10.0.0.4 0
link bi 10.0.0.1 10.0.0.5 0
link bi 10.0.0.1 10.0.0.6 0

link bi 10.0.0.2 10.0.0.6 0

link bi 10.0.0.3 10.0.0.4 0
link bi 10.0.0.3 10.0.0.5 0

link bi 10.0.0.4 10.0.0.6 0

```

Figure 8.5: `thrp_switch` commands used to create network topology for Figure 8.1

8.6 Future work

The current C implementation of TH(O)RP is in early stages of development and still requires some testing. It is also not as complete as the C++ implementation developed for the simulations. Future improvements planned for the C implementation are to include the delivery system, which include ACKs and retransmissions, to improve robustness. It is also planned to fully implement the plug-in interface, to make the implementation more expandable.

Other future developments would include expanding the current implementation of multiple interface declarations (MID), and to provide Internet connectivity through the host and network association (HNA) messaging. It is also planned to develop a parent selection scheme that would not rely on a manually defined central entity, and that could automatically derive a hierarchy from the network topology. This scheme would resemble the method employed in the OrderOne Networks protocol [29], which uses a direct sequence distance vector (DSDV) algorithm to calculate routes to all nodes in the network, and then the neighbour node through which the greatest number of destinations is reachable, is chosen as the parent. This implementation might not be entirely relevant to the target communication network, as it could place the centre of the network at a different location than the central SCOOT controller, from where all TICs need to be monitored. This could bring additional latencies that would be undesirable. By implementing an alternative parent selection scheme, however, could open up new implementations where TH(O)RP could be employed.

A security plug-in would definitely be on the list of future developments, as wireless networking security can be weak. In order to use TH(O)RP as a routing protocol, the routing information of the Linux kernel, or the kernel routes, would need to be manipulated. Future versions of TH(O)RP would be made to communicate with the kernel, so that the routing information it collects can be made available to applications.

With the rapid growth of the Internet and the number of connected devices, the current pool of available IPv4 addresses is fast running out. To remedy this shortcoming the IPv6 protocol and addressing was created. The current version of TH(O)RP does have mechanisms in place to accommodate IPv6 addresses, but it has not yet been fully implemented. Automatic IP address configuration, used to automatically assign unique IP addresses to all TH(O)RP nodes in a network, would also be included in future developments.

Finally, it would be considered to make TH(O)RP compatible with Windows systems, to further expand its portability and increase the range of applications in which it can be used.

8.7 Conclusion

At present the C implementation is relatively basic, but much of the infrastructure has been developed. With a C++ implementation that can run in a simulation environment, new methods can easily be developed, and additional functionality can be added to the TH(O)RP protocol. These functions can then be easily translated to C and included in the C implementation of TH(O)RP, which can be used in real networking environments.

Chapter 9

Conclusion

9.1 Overview of work

The need for communication infrastructure to deploy ITS systems

Traffic networks around the world are becoming more congested with each passing year. To remedy the problem, many cities have deployed ITS systems to manage their transport networks more efficiently.

In third world countries, the communication infrastructure required to deploy large scale ITS systems are often lacking or non-existent. Due to strong economic growth experienced in South Africa in recent years, South African roads have become chronically congested. Subsequently, South African cities started to deploy ITS systems to manage their traffic networks. However, many cities still lack sufficient communication infrastructures to deploy these systems.

Wireless networks continue to expand in their capacity and applications, and they could play a crucial role in solving the infrastructure problems faced by many cities and municipalities.

Project objectives and outcomes

The project had four main objectives. Through the achievements of these objectives a number of contributions were also realised:

1. A study on alternative communication solutions that could provide the infrastructure required to deploy ITS systems was conducted.
 - In Chapter 2, the requirements of such a telecommunication network were determined, and various communication technologies investigated.
 - In Section 2.3, a conceptual communication network was formulated, and a network architecture was suggested.
2. The design and development of an ad hoc wireless network routing protocol, that could realise the communication network proposed as part of the first project objective.
 - In Chapter 3, a study on a few of the most popular ad hoc routing protocols was conducted, and the strengths and weaknesses of these protocols highlighted.
 - In Chapter 4, the TH(O)RP routing protocol was conceptualised from the knowledge gained from investigating other ad hoc protocols, and additional functionality was added to optimise the behaviour of the protocol, making it more stable and robust.

3. In Chapter 5, network latency models were created that considered different communication technologies and modeled the operation of the TH(O)RP routing protocol.
 - These models could estimate the latencies experienced by different communication technologies, when running the routing protocol over various single- and multi-hop scenarios.
 - The models could also derive the minimum bandwidth required to successfully operate the TH(O)RP protocol.
4. The routing protocol was evaluated and optimised through simulation.
 - Chapter 6 described the OMNET++ simulation environment used to simulate the protocol, and how the protocol was incorporated into the simulation. It also discussed the various options that can be set for the protocol, and what output is available during simulation. Finally, the various simulation scenarios derived to test the protocol, were discussed.
 - In Chapter 7 the results of the tests are discussed, and various optimisations are implemented in the protocol.
 - As a consequence of the simulation results and the way in which TH(O)RP works, it was attempted as a further initiative to provide methods that could make the protocol more responsive and robust towards network failures, namely:
 - Increased parent selection interval: A node would increase its update intervals in the event of losing its parent, allowing rapid discovery of new parent candidates.
 - Adoption announcement: A node will advertise that it can be selected as a parent as soon as it selects a parent, allowing surrounding nodes to select it as a potential parent more rapidly.
 - Assisted HSPP: A modified RREQ message is issued when a node selects a new parent, actively informing the parent that it is a new child.
 - Route error notification: A message that is issued to nodes lower down the hierarchy from a node that loses its parent and is unable to select a new parent. This allows nodes lower down the hierarchy to look for alternative parents more promptly.
 - Cumulative RREQ: A node that has children can issue a CRREQ to update the routes of all the nodes beneath it when it selects a new parent, stabilising the network hierarchy.
 - The results of the simulations proved the value of a hierarchical approach in large networks, and showed that localising the effects of network topology changes helps reduce the bandwidth requirements of ad hoc routing protocols.
 - As a result of developing the simulation, a tool was developed that could be used as a platform for developing new methods for TH(O)RP and testing them in various network topologies.
5. An additional project objective, not previously envisioned, was achieved through the development of a workable implementation of the TH(O)RP routing protocol, which was

written in C. The implementation discussed in Chapter 8, is however, in the early stages of development and not yet fully featured. Regardless, it provides a lightweight implementation, with few dependencies, that could be used in network environments, or in embedded hardware, to evaluate the routing protocol in real-life situations.

9.2 Further work

Additional routing protocol functionality

In Chapter 4, a few improvements were mentioned that could be added to the TH(O)RP routing protocol. The communication network proposed in Chapter 2 used a mixed technology solution, with a fibre optic backbone, and Wi-Fi and WiMAX radios. Therefore, each node of the network could potentially have multiple network interfaces. The addition of multiple interface deceleration (MID) messaging would allow the protocol to utilise multiple network interfaces.

Ad hoc networks will almost always be connected to the Internet at some point, and the inclusion of host and network association (HNA) messaging would allow the protocol to identify gateway nodes, and allow applications using the protocol to connect to the Internet. Due to the security risks associated with wireless networks, the routing protocol would need to be secured. Methods for securing communication and controlling access to networks running the routing protocol, could be investigated and implemented.

The tree hierarchy derived by the routing protocol is automatically derived from the network topology. This could result in trees that are poorly balanced, which could put unnecessary strain on a few nodes high up in the tree hierarchies. By utilising nodes willingness, certain nodes can be made more attractive options for selection as parents. This could be used to define a few dedicated nodes throughout the network that could provide a reliable backbone, and could be used to balance the resulting tree hierarchy.

Because any communication network often runs multiple services, it is necessary to provide a method for prioritising certain services over others. Further work can be conducted in providing quality of service (QoS) in ad hoc wireless routing protocols, and how these can be incorporated into the TH(O)RP protocol.

IP addressing requires each node in a network to have a unique address. Because ad hoc networks are automatically configured, a mechanism is required that could ensure that each node in the network has a unique IP address. The routing protocol developed could be improved by introducing mechanisms that could achieve automatic address configuration.

Research areas identified during simulation

In its current state, the simulation developed in Chapter 6 only considers nodes with single network interfaces. The simulations could be expanded, by adding situations where multiple network interfaces are used by nodes, and evaluating the effects.

The TRAFX protocol and SCOOT systems require accurate timing, within tight parameters. In order to achieve accurate synchronisation of system clocks at all the TICs, a time server would be required. The current simulation does not include time servers, and further work could be done into incorporating a time server in the simulation and evaluating its performance.

During the simulations it was found that due to the periodic nature of the routing protocol, it struggled to identify network failures in a timely fashion. Although methods were developed to speed up the recovery from errors, the discovery of network failures still requires further research. It was also found that in order to make the protocol more robust, redundancy would need to be included. It was proposed that redundancy could be introduced by maintaining multiple routes, or by maintaining multiple tree hierarchies.

Because of the static nature of the communication infrastructure, the location of nodes tend to remain constant. Therefore, the periodic updates generated by the protocol can be adjusted as the network stabilises. Efficient mechanisms for adjusting the periodic updates can be further researched.

The best effort delivery system implemented in the routing protocol helps in stabilising the tree hierarchy and provides more reliable delivery of data messages. One issue that came to light during simulation, was that retransmitted messages suffered an unnecessary time penalty by being subjected to the forwarding jitter upon each retransmission. More efficient methods of handling retransmissions could be investigated.

Originally, link states and link hysteresis were used to construct routes around symmetric links and along the least number of hops. With the introduction of the ETX link- and WCETX route metrics, which measures the forward and reverse link loss probabilities, the link states and link hysteresis have become obsolete. This could be removed in future versions of the TH(O)RP protocol.

As mentioned before, the proposed communication network architecture utilises multiple communication technologies, and could be configured with multiple radios. In these situations, the different network interfaces will most probably have different bandwidths. The current link metric will not be able to distinguish between low and high bandwidth links, and the ETT link- and WCETT route metrics could be introduced to establish more efficient routes along high bandwidth paths.

When a subtree is disconnected from the main tree hierarchy and is unable to reestablish a connection, it will gradually deteriorate. Methods into persistent subtrees could be investigated, and how nodes in an isolated subtree could register that they are disconnected from the main tree, and how they can reestablish a tree hierarchy amongst themselves.

The tree hierarchy used by the protocol to segment the network, and to achieve efficient routing without any flooding, highlighted the benefits that a tree structure could add to achieving efficient ad hoc routing protocols. More research could be conducted into optimising these trees, and how they can be used to address other problems faced by ad hoc routing protocols.

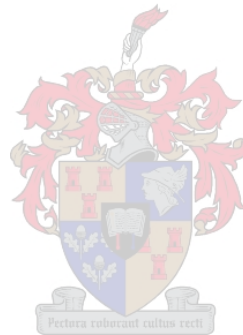
Extending the functionality of the C working version

As mentioned in Chapter 8, the current C implementation of the routing protocol is in an early development stage. Adding all of the functionality included in the C++ implementation used during simulation, would make the C implementation more complete. The C implementation can also be programmed into hardware, to test the TH(O)RP protocol in a real-life network environment. The C implementation could also be made compatible with IPv6 addressing.

9.3 Summary

The objectives of the project were achieved, with the project culminating in the successful proof of concept, of the applicability of ad hoc wireless networks to provide a communication infrastructure capable of facilitating ITS systems.

In achieving the project objectives, the envisioned contributions were also realised. Firstly, a summary of different communication technologies and their applicability to facilitate ITS systems was compiled. Unfortunately, due to a breakdown of cooperation between the University of Stellenbosch and the CoCT, the findings of the feasibility study could not be presented to the CoCT. Secondly, an ad hoc routing protocol was successfully simulated, and new methods were developed to improve the stability of the protocol. The simulation also provides a good platform for future research and development, as new concepts can be easily incorporated into the protocol and evaluated. Finally, contributions were made to the ongoing research and development of ad hoc wireless routing protocols, by developing methods that can be used to improve the scalability and stability of these protocols. Contributions were also made to the ongoing development of open source routing protocols by releasing the software developed in this project under the GPL.



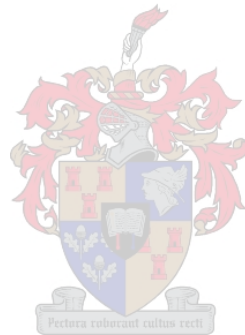
Bibliography

- [1] "866-00283-ACTMTC PTLC Requirement Specification, Plessey Tellumat SA." 1990.
- [2] "3G." <http://en.wikipedia.org/wiki/3g>. June 2006.
- [3] "802.16." http://isp.webopedia.com/TERM/8/802_16.html. June 2006.
- [4] "Bandwidth." http://www.answers.com/topic/bandwidth#after_ad3. June 2006.
- [5] "Define 3G - a definition from Whatis.com."
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci214486,00.html. June 2006.
- [6] "Enhanced Data Rates for GSM Evolution." <http://en.wikipedia.org/wiki/EDGE>.
June 2006.
- [7] "General Packet Radio Service." <http://en.wikipedia.org/wiki/GPRS>. June 2006.
- [8] "Goal Technology Solutions." <http://www.goal.co.za/>. November 2006.
- [9] "Google images." http://www.claranet.fr/img/grfx_wifi.gif. June 2006.
- [10] "Google images." <http://www.pec-forum.com/img/2004/wimax.gif>. June 2006.
- [11] "High-Speed Downlink Packet Access." <http://en.wikipedia.org/wiki/HSDPA>. June 2006.
- [12] "List of device bandwidths." <http://www.answers.com/main/list-of-device-bandwidths>.
June 2006.
- [13] "Networking definitions - Wi-Fi."
http://searchnetworking.techtarget.com/sDefinition/0,,sid7_gci838865,00.html. June 2006.
- [14] "Optical fiber." http://en.wikipedia.org/wiki/Optical_fibre. June 2006.
- [15] "PCMAG.com - The independent guide to technology - 3G."
http://www.pcmag.com/encyclopedia_term/0,2542,t=3G&i=55406,00.asp. June 2006.
- [16] "Power line communication." http://en.wikipedia.org/wiki/Power_line_communication.
November 2006.
- [17] "Sydney coordinated adaptive traffic system." <http://en.wikipedia.org/wiki/SCATS>.
November 2006.

- [18] "Vodacom South Africa." <http://www.vodacom.co.za/services/>. July 2006.
- [19] "Wi-Fi." <http://en.wikipedia.org/wiki/Wi-fi>. June 2006.
- [20] "WiMAX." <http://en.wikipedia.org/wiki/Wi-max>. June 2006.
- [21] AKYLDIZ, I. and WANG, X., "A Survey on Wireless Mesh Networks." *IEEE Radio Communications*, September 2005, pp. 23–30.
- [22] ALMERTH, K. and CHALMERS, R., "A mobility gateway for small device networks.." *Technical report*, March 2004.
- [23] BELDING-ROYER, E., PERKINS, C., and SUN, Y., "Internet connectivity for ad-hoc mobile networks.." *Technical report*, April 2004.
- [24] CCITT, "Yellow book v23 specification." Volume VIII - Fascicle VIII.1, VIIIth plenary assembly, November 1980.
- [25] CCITT, "Cape town traffic control system trafox comms link: Installation to controllers interface control document." ADV10301.WPF, December 1993.
- [26] CLAUSEN, T. and JACQUET, P., "Optimized link state routing protocol (OLSR)." *Request for Comments: 3626*, October 2003.
- [27] CORSON, S. and MACKER, J., "Mobile ad hoc networking (MANET): Routing protocol performance issues and evaluation considerations." *Request for Comments: 2501*, January 1999.
- [28] COUTO, D. S. J. D., *High-Throughput Routing for Multi-Hop Wireless Networks*. PhD thesis, Massachusetts Institute of Technology, MIT Department of Electrical Engineering and Computer Science, June 2004.
- [29] DAVIES, D. and DAVIES, C., *An Abbreviated Technical Discussion of the OrderOne Networks Protocol, Version 1.0*.
- [30] DRAVES, R., PADHYE, J., and ZILL, B., "Comparison of Routing Metrics for Static Multi-Hop Wireless Networks." *Microsoft Research*, September 2004.
- [31] DRAVES, R., PADHYE, J., and ZILL, B., "Routing in Multi-Radio, Multi-Hop Wireless Mesh Networks." *Microsoft Research*, 2004.
- [32] FOR PERVASIVE COMPUTING, N. H., May 2005.
- [33] GANSNER, E., KOUTSOFIOS, E., and NORTH, S., "Drawing graphs with dot." February 2002.
- [34] GERLA, M., KULKARNI, G., NANDAN, A., and SRIVASTAVA, M., "A Radio Aware Routing Protocol for Wireless Mesh Networks." *UCLA Electrical Engineering, UCLA Computer Science*, 2004.
- [35] GOODRICH, M. T. and TAMASSIA, R., *Data structure and algorithms in JavaTM*. Second edition. John Wiley & Sons, Inc., 2001.

- [36] HAAS, Z. J., "A New Routing Protocol for the Reconfigurable Wireless Networks." *School of Electrical Engineering, Cornell University*, 1997.
- [37] JURDAK, R., LOPES, C. V., and BALDI, P., "A Survey, Classification and Comparative Analysis of Medium Access Control Protocols for Ad Hoc Networks." *IEEE Communications Surveys*, First quarter 2004, Vol. 6, No. 1, pp. 2–16.
- [38] LOBBERS, M. and WILLKOMM, D., *A Mobility Framework for OMNeT++*, User Manual. Version 1.0a4 edition. Telecommunication Networks Group at the Technische Universitaet Berlin, June 2004.
- [39] MURTHY, C. S. R. and MANOJ, B., *Ad Hoc Wireless Networks: Architectures and Protocols*. Prentice Hall Communications Engineering and Emerging Technologies Series, May 2004.
- [40] OGIER, R., TEMPLIN, F., and LEWIS, M., "Topology dissemination based on reverse-Path forwarding (TBRPF)." *Request for Comments: 3684*, February 2004.
- [41] PERKINS, C., BELDING-ROYER, E., and DAS, S., "Ad hoc on-demand distance vector (AODV) routing." *Request for Comments: 3561*, July 2003.
- [42] PERKINS, C. and BHAGWAT, P., "Highly dynamic destination-sequence distance vector routing (DSDV) for mobile computers." *SIGCOMM*, 1994.
- [43] PETERSON, L. L. and DAVIE, B. S., *Computer Networks - A Systems Approach*. Second edition edition. Morgan Kaufmann publishers, 2000.
- [44] SCHROEDER, A. and AMANNA, A., *On the Road with Wireless Network - A roadmap of Wireless Network Issues for Transportation Professionals*. Virginia Tech, Transportation Institute, Center for Technology Deployment, May 2005.
- [45] TANENBAUM, A. S., *Computer Networks*. Fourth edition. Pearson Education Inc., Prentice Hall PTR, 2003.
- [46] TØNNESEN, A., "Impementing and extending the Optimized Link State Routing Protocol." Master's thesis, University of Oslo, Department of Informatics, 2004.
- [47] TØNNESEN, A., "Olsr_switch network simulation."
<http://www.olsr.org/doc/README-Olsr-Switch.html>. 2004.
- [48] VARGA, A., *OMNeT++*, *Discrete Event Simulation System*, User Manual. Version 3.2 edition. Technical University of Budapest, March 2005.
- [49] WOLHUTER, R., *Elements of Telecommunications Systems Design and Teletraffic Analysis*. Centre for Electrical and Electronic Engineering, University of Stellenbosch, 2005.

Appendices



Appendix A

Packet formats and headers

A.1 TH(O)RP packet and message formats

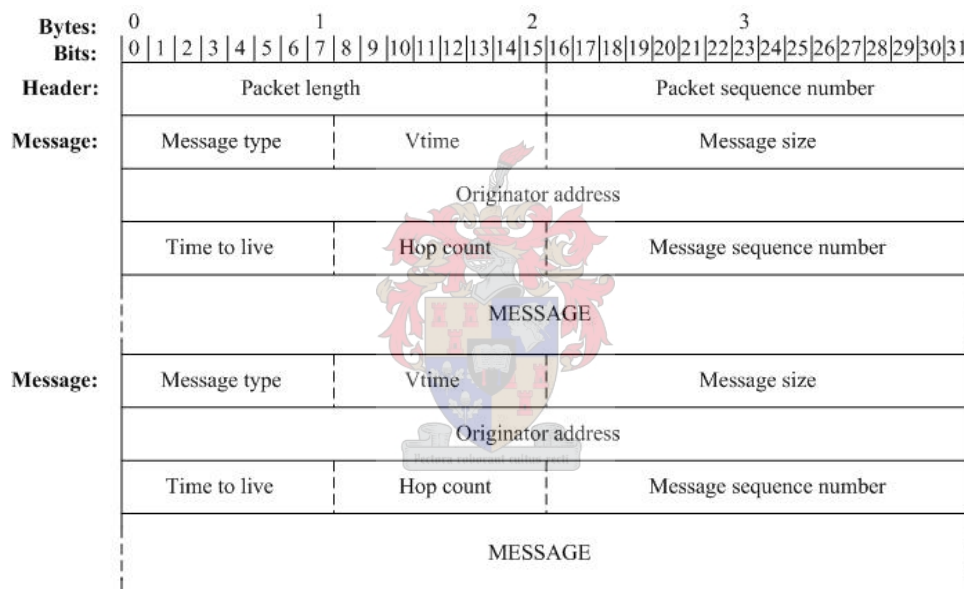
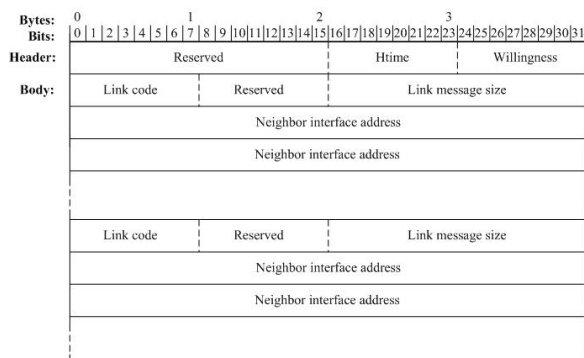
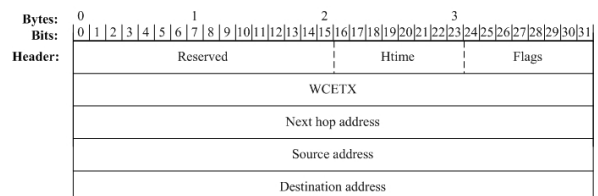


Figure A.1: TH(O)RP packet- and message format

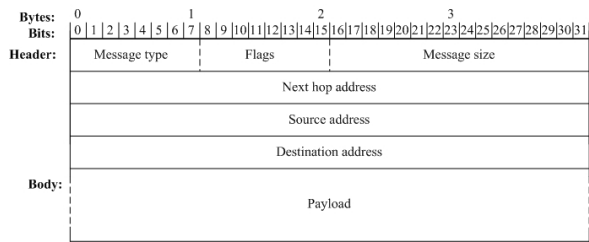


(a) HELLO message format

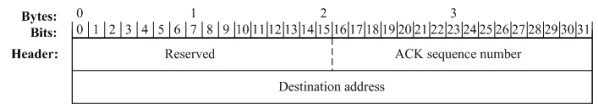


(b) Route request message format

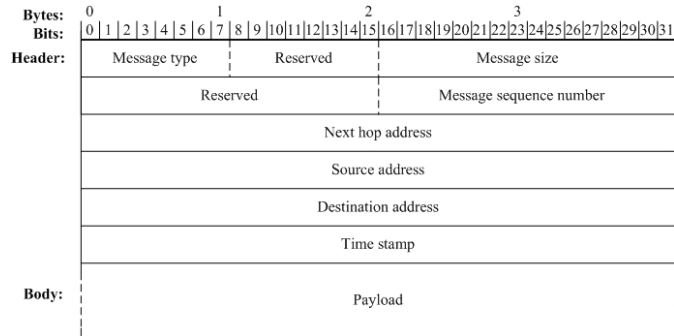
Figure A.2: The HELLO and RREQ messages



(a) Data message format



(b) ACK message format



(c) TRAFX message format

Figure A.3: Data, ACK and TRAFX messages

A.2 Miscellaneous routing, transport and MAC layer packet and frame formats

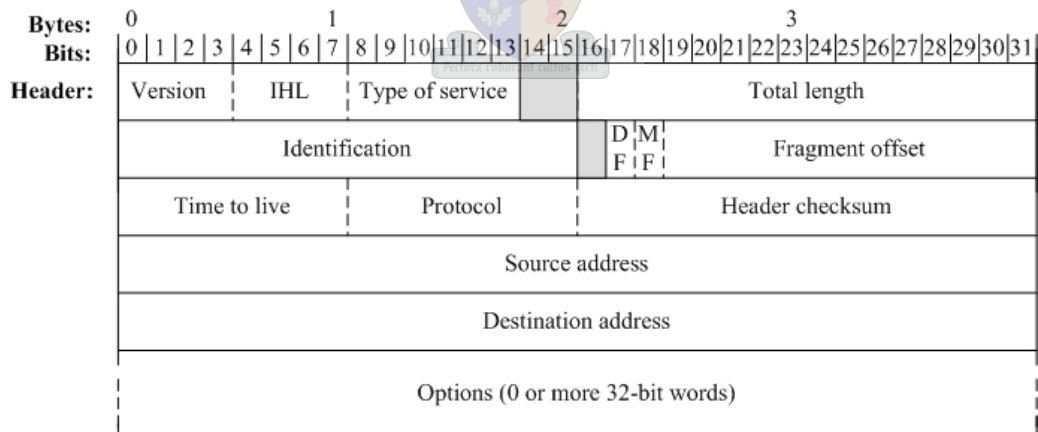


Figure A.4: IPv4 packet format

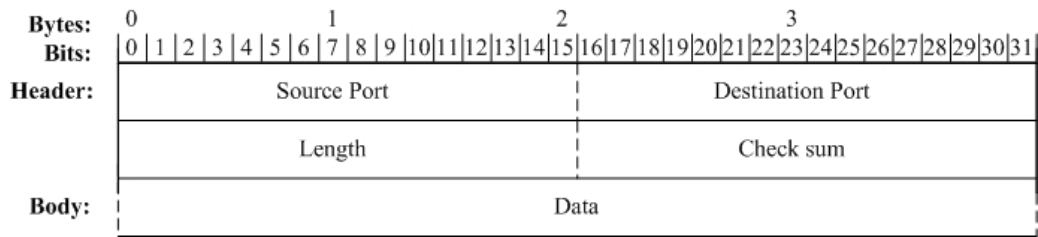


Figure A.5: UDP packet format

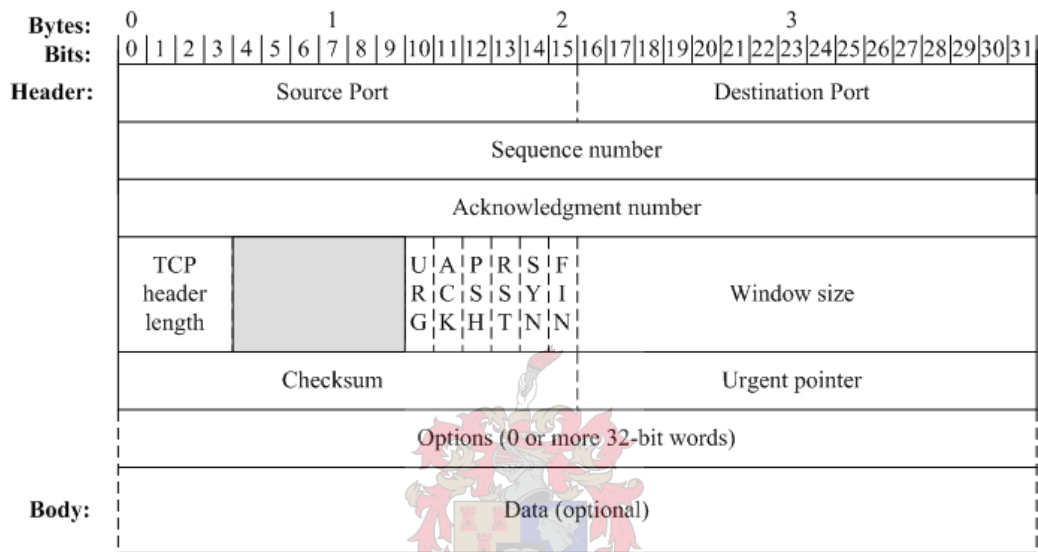


Figure A.6: TCP packet format

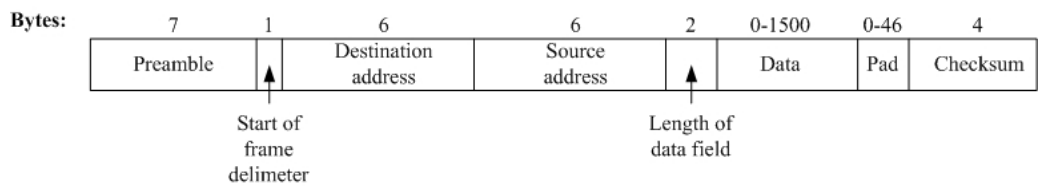


Figure A.7: Ethernet (802.3) packet format

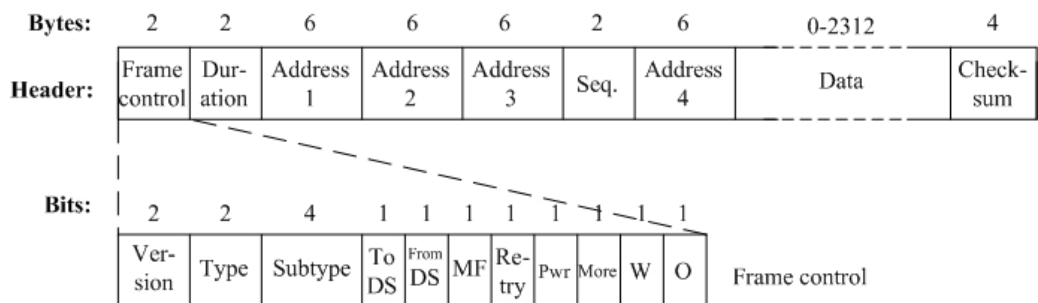


Figure A.8: Wi-fi (802.11) packet format

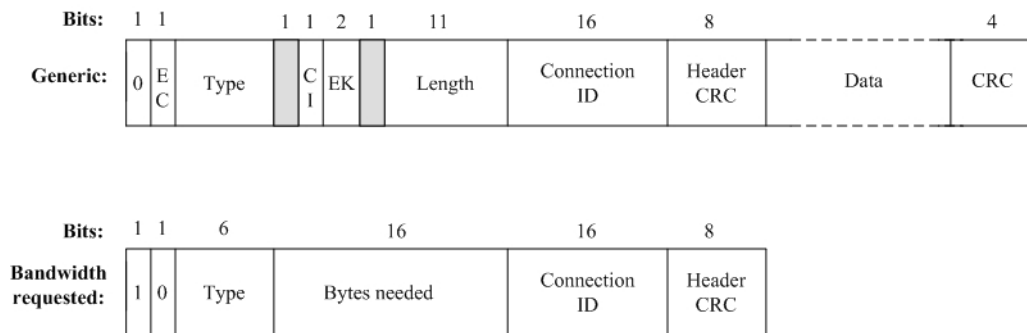


Figure A.9: WiMAX (802.16) packet format: Generic and bandwidth requested

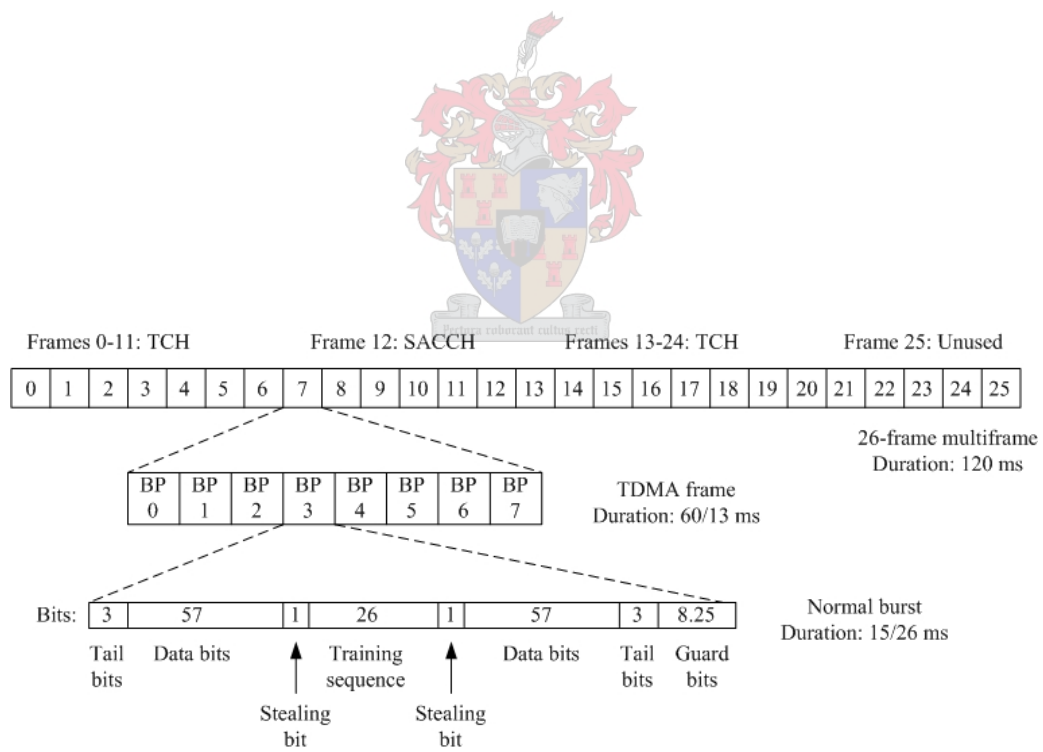


Figure A.10: GSM packet format

Appendix B

TH(O)RP message definitions in OMNET++ simulation

```

//*****
// * file:      ThrpPkt.msg
// *
// * author:    Daniel Weich Morrison
// *
// * copyright: (C) 2006 Digital Signal Processing Group (DSP) at
// *           Stellenbosch University, South Africa.
// *
// *           This program is free software; you can redistribute it
// *           and/or modify it under the terms of the GNU General Public
// *           License as published by the Free Software Foundation; either
// *           version 2 of the License, or (at your option) any later
// *           version.
// *           For further information see file COPYING
// *           in the top level directory
// *****
// * part of:   framework implementation developed by tkn
// * description: - Define the TH(O)RP messages here in OMNET++ .msg format.
// *****
// * changelog: $0.1$
// *           last modified:  $Date: 2006-07-24$
// *           by:             $Author: dw-morrison $
// *****/

// # Don't forget to include the header file of the basic msg
// This defines the message base class, from where all other messages inherit
// certain base functions.
cplusplus {
#include <ApplPkt_m.h>
#include <NetwPkt_m.h>

// Scaling factor
//////////
#define VTIME_SCALE_FACTOR 0.0625          // Valid time scaling factor

// Emission Intervals
//////////
#define HELLO_INTERVAL      2              // Hello message emission interval
#define REFRESH_INTERVAL   2              // Unused (N/A)
#define PARENT_SEL_INTERVAL 3 * HELLO_INTERVAL // Parent selection/update interval
#define RREQ_SELF_INTERVAL PARENT_SEL_INTERVAL // Route update interval
#define MID_INTERVAL       5              // Mid message emission interval
#define HNA_INTERVAL       MID_INTERVAL   // Hna message emission interval

```

```

// Holding/Validity Time
//
//
#define NEIGHB_HOLD_TIME    3 * HELLO_INTERVAL // Hello message hold/valid interval
#define MID_HOLD_TIME      3 * MID_INTERVAL    // Mid message hold/valid interval
#define HNA_HOLD_TIME      3 * HNA_INTERVAL    // Hna message hold/valid interval
#define ROUTE_HOLD_TIME    5 * RREQ_SELF_INTERVAL // Route hold/valid interval

// Link Hysteresis
//
//
#define HYST_THRESHOLD_HIGH 0.8 // Hysteresis high threshold
#define HYST_THRESHOLD_LOW 0.3 // Hysteresis low threshold
#define HYST_SCALING        0.5 // Hysteresis scaling constant

// Misc. Constants
//
//
#define MAXJITTER           HELLO_INTERVAL / 16 // Maximum forwarding jitter constant
#define MAX_TTL             0xff // Maximum TTL constant
// Seqnos are 16 bit values
#define MAXVALUE            0xFFFF // Maximim sequence number constant

// Link Types
//
//
#define UNSPEC_LINK        0 // Unspecified link
#define ASYM_LINK          1 // Asymmetric link
#define SYM_LINK           2 // Symetric link
#define LOST_LINK          3 // Lost link
#define HIDE_LINK          4 // Hidden link
#define MAX_LINK           4 // Maximum link value

// Neighbour Types
//
//
#define NOT_NEIGH          0 // Not a neighbour
#define SYM_NEIGH          1 // Symetric neighbour
#define PARENT_NEIGH      2 // Neighbour is my parent (N/A)
#define CHILD_NEIGH       3 // Neighbour is my child (N/A)
#define ROOT_NEIGH        4 // Neighbour is ROOT of tree hierarchy (N/A)
#define WiMAX_NEIGH       5 // Neighbour is a WiMAX node (N/A)
#define MAX_NEIGH         5 // Maximum neighbour value

// Neighbour status
//
//
#define NOT_SYM            0 // Neighbour is not symmetric
#define SYM                1 // Neighbour is symmetric

// Willingness
//
//
#define WILL_NEVER         0 // N/A
#define WILL_LOW           1 // N/A
#define WILL_DEFAULT      3 // N/A
#define WILL_HIGH         6 // N/A
#define WILL_ALWAYS       7 // N/A
#define WILL_NEW_PARENT   8 // Used to accomodate fast recovery and AHSPP

// Message types
//
//
#define HELLO              1 // LQHello message
#define MID                2 // MID (multiple interface declaration) message
#define HNA                3 // HNA (host and network association) message
#define RREQ               4 // RREQ (route request) message
#define CRREQ              5 // CRREQ (cumulative route request) message
#define RREP               6 // RREP (route reply) message
#define RERR               7 // RERR (route error) message (N/A)

```



```

#define ROPT          8          // ROPT (route optimize) message (N/A)
#define DATA        9          // DATA message
#define ACK          10         // ACK (acknowledgment) message
#define MAX_MSG      10         // The maximum message value

// Scheduler parameters
////////////////////////////////////
#define POLL_RATE    0.05       // The poll-rate of the scheduler
// (timeout events will happen each poll)

// Hysteresis parameters
////////////////////////////////////
#define hscaling     HYST_SCALING
#define hhigh        HYST_THRESHOLD_HIGH
#define hlow         HYST_THRESHOLD_LOW

// Link Quality parameters
////////////////////////////////////
#define INFINITE_ETX ((float)(1 << 30)) // Approximating infinity for etx value
#define MIN_LINK_QUALITY 0.01         // The minimum link quality allowable

// Route setup parameters
////////////////////////////////////
#define AHSPP_ON      2          // Indicate this is a HSPP rreq (0b00000010)
#define AHSPP_OFF    253        // Indicate this is not a HSPP rreq (0b11111101)

// Duplicate set parameters
////////////////////////////////////
#define DUP_HOLD_TIME 30         // The duplicate message hold time

// Retry table parameters
////////////////////////////////////
#define RETRY_TIME    MAXJITTER*1.5 // The retry timeout time
#define RETRY_ON      1          // The retry on flag (0b00000001)
// (used to indentify retried messages)
#define RETRY_OFF     254        // The retry off flag (0b11111110)
#define RETRIES       4          // The number of retry attempts

// Parent selection parameters
////////////////////////////////////
// Selection methods
#define P_CENTRAL_C    0          // Central parent selection method (default)
#define P_NONCENTRAL_C 1         // Non-central selection method (N/A)
// (network has to derive center of network)

// Constants
#define NO_PARENT     -1         // Indicate that node does not have a parent
#define ROOT_ROOT_COST 0        // Parameter for the route cost of the ROOT
#define MAX_CHILDREN  32        // Maximum number of children allowed

// Route error
////////////////////////////////////
#define PARENT_CONN_LOST 0       // Indicates that a route error occurred due
// to lost parent connection
#define PARENT_CONN_FIXED 1      // Indicates that a new connection to parent
// was established by sending node

// Application layer paramaters
////////////////////////////////////
// Message types
#define TRAFX_REQ      0          // TRAFX request
#define TRAFX_REP      1          // TRAFX reply
#define TRAFX_CSV      2          // TRAFX message with comma seperated data

```

```

// Conversation types
#define POLY          0          // Polylogue conversation type (one-to-one)
#define DIA          1          // Dialogue conversation type (one-to-many)

// Application layer msg forwarding scheme
#define IMMEDIATE    0          // Data messages are forwarded immediately
#define DELAYED      1          // Data messages are buffered and forwarded
                                // at next forwarding event

// Utility variables
////////////////////////////////////
#define CONSOLE_SIZE 30        // Utility variable for clearing console screen
#define P_DATA       1        // Print data output bubble msgs (0b00001)
#define P_ACK        2        // Print ack output bubble msgs (0b00010)
#define P_ROUTE      4        // Print route output bubble msgs (0b00100)
#define P_MID_HNA    8        // Print mid/hna output bubble msgs (0b01000)
#define P_HELLO      16       // Print hello output bubble msgs (0b10000)
};

// the class of the basic msg has to be declared:
class NetwPkt;

// template file for creating own messages
//
// own .msg files have to be derived from the obligatory basic .msg files
// (see manual)
// Replace YourPkt with the name you want the msg to have and
// replace BasePkt with the name of the corresponding obligatory
// basic message
//
// @author Daniel Weich Morrison

//*****
/** BROADCAST MESSAGES */
//*****

// LQ Neighbour information.
// Information about the nodes visible to this node.
////////////////////////////////////
struct neighInfo
{
    fields:
        int      linkType;      // The type of link between node and a neighbour
        int      neighType;     // The type of neighbour this is
        double   linkQuality;    // The link quality as seen from this node (p_f)
        double   neighLinkQuality; // The link quality as seen from a neighbour node (p_r)
        int      addr;          // The IP address of the neighbour node
};

// LQ HELLO message defintion.
// The Hello message is used to facilitate neighbour discovery and link sensing.
// It gathers and publishes information about the one hop neighbourhood of each node.
////////////////////////////////////
class lqHelloMsg
{
    fields:
        /*# Define the LQ_Hello fields here
        double htime;          // The hold time for this message
        int    will;           // The willingness of this node (N/A)
        int    haveparent;     // Specify whether this node has a parent
        int    numchild;       // Specify the number of children this node has
        double rootcost;       // Specify the route cost to the ROOT of network
        */
};

```

```

    int    paddr;          // Specify the IP address of the parent of this node

    /*# Add the neighbour info
    neighInfo neigh[];    // Information about the neighbourhood of this node
};

// Multiple Interface Declaration (MID) message definition.
// The message used to carry aliases of nodes around the network.
// We define aliases for nodes if they have multiple network interfaces
// and we want to uniquely identify this node across each interface.
////////////////////
struct midAddr
{
    fields:
        int addr;          // Alias IP address of this node
};
class midMsg
{
    fields:
        midAddr mid_addr[]; // Multiple MID addresses can be contained in this message
};

// Host and Network Association (HNA) message.
// The message used to carry information about gateways to the Internet
// around the network.
////////////////////
struct hnaPair
{
    fields:
        int  addr;          // IP address of a gateway node
        int  netmask;       // The netmask of a gateway
};
class hnaMsg
{
    fields:
        hnaPair hna_net[]; // Multiple HNA pairs can be contained in this message
};

//*****
/* UNICAST MESSAGES */
//*****
// The Route request (RREQ) message.
// The message used to request a route to a destination (usually ROOT of network).
////////////////////
class rreqMsg
{
    fields:
        char  flags;        // Flags for this message (retry flag)
        int   htime;        // The hold-time for this message
        double wcost;       // The cost of the route in WCETX metric
        int   next_hop;     // The IP address of the next hop node
        int   src;          // The IP address of the source node
        int   dest;         // The IP address of the destination node
};

struct subHier
{
    fields:
        int   addr;         // Destination address
        int   hop_cnt;      // Hop count to destination
        double wcost;       // WCETX metric for route to destination
        double timer;       // Time out for destination
};

```

```

// The cummalative Route request (CRREQ) message.
// The message used to request a route to a destination (usually ROOT of network).
//////////
class cRreqMsg
{
    fields:
        char    flags;        // Flags for this message (retry flag)
        int     htime;        // The hold-time for this message
        double  wctx;         // The cost of the route in WCETX metric
        int     next_hop;     // The IP address of the next hop node
        int     src;          // The IP address of the source node
        int     dest;         // The IP address of the destination node
        subHier dests[];     // List of destinations lower in hierarchy
};

// The Route reply (RREP) message.
// The message used to notify a node that a route has been succesfully setup,
// along with the cost of the route and other information.
//////////
class rrepMsg
{
    fields:
        char    flags;        // Flags for this message (retry flag)
        int     rrepSeqNum;   // The sequence number of the RREQ message replying too
        int     htime;        // The hold-time for this message
        int     status;       // The status of the route (N/A)
        double  wctx;         // The cost of the route in WCETX metric
        int     next_hop;     // The IP address of the next hop node
        int     src;          // The IP address of the source node
        int     dest;         // The IP address of the destination node
};

// The Route error (RERR) message (N/A).
// The message used to notify nodes along a route that the route has vailed.
//////////
class rerrMsg
{
    fields:
        int     err_cause;    // Specify the cause of the error
        int     next_hop;    // IP address of the next hop node
        int     src;         // IP address of the source node
        int     dest;        // IP address of the destination node
};

//*****
//*  DATA MESSAGES  */
//*****

// The Data message.
// Message used to carry data around the network.
// (payload -> will be an encapsulated TRAFX application layer packet).
//////////
class dataMsg
{
    fields:
        int     msgType;     // The type of the encapsulated message
        char    flags;       // Flags for this message (retry flag)
        int     msgSize;     // The size of this message
        int     next_hop;    // The IP address of the next hop node
        int     src;         // The IP address of the source node
        int     dest;        // The IP address of the destination node
        //# payload -> will be an encapsulated TRAFX application layer packet
};

```

```

};

/** The ACK message (obsolete)
//////////
**/class ackMsg
/**{
/** fields:
/** int ackSeqNum;
/** char flags;
/** int msgSize;
/** int next_hop;
/** int src;
/** int dest;
/** // payload -> will be an encapsulated TRAFX application layer packet
/**};

// The ACK message.
// Acknowledges data messages or routed messages
// on final hop (see "indirect acknowledgments").
//////////
class ackMsg
{
    fields:
        int ackSeqNum; // The sequence number of the message we are acknowledging
        int dest; // The IP address of the destination node
};

// The generic THRP message (THRP common).
// Can only contain one (LQHello, HNA, ...) message at a time.
//////////
class ThrpMsg
{
    fields:
        /** THRP message header
        //////////
        int msgType; // Thrp message type (LQHello, HNA, ...)
        double vTime; // The valid time parameter for this message
        int msgSize; // Size of this message
        int originator; // IP address of originator of this message
        int ttl; // The TTL (time to live) for this message
        int hopCnt; // The hop count of this message
        int msgSeqNum; // The sequence number of this message
        /** THRP message body
        //////////
        lqHelloMsg lqHello[]; // LQHello message placeholder
        rreqMsg rreq[]; // RREQ message placeholder
        crreqMsg crreq[]; // CRREQ message placeholder
        rrepMsg rrep[]; // RREP message placeholder
        rerrMsg rerr[]; // RERR message placeholder
        midMsg mid[]; // MID message placeholder
        hnaMsg hna[]; // HNA message placeholder
        dataMsg data[]; // Data message placeholder
        ackMsg ack[]; // ACK message placeholder
        /**struct roptMsg[];
};

// Generic THRP packet.
// Can contain multiple thrp messages (allow piggybacking).
//////////
message ThrpPkt extends NetWPkt {
    properties:
        customize = true;
    fields:

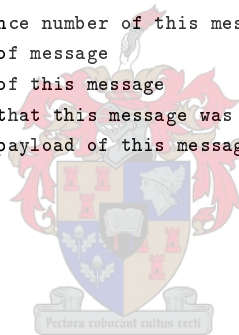
```

```
// Have a stack of THRP messages
abstract ThrpMsg msg[];
};

//////////
// Application level network messages
//////////
//# the class of the basic msg has to be declared:
class ApplPkt;

// A simple struct used to "emulate" a payload for this package.
class payLoad
{
    fields:
        //# Define the payload fields here.
        char    content[100];    // The data payload of this package
};

// Generic Application level package (TRAFX)
// Messages used to facilitate the TRAFX protocol of the CoCT.
//////////
message TrafXpkt extends ApplPkt {
    properties:
        customize = false;
    fields:
        int    msgSeqNum;        // The sequence number of this message
        int    msgType;         // The type of message
        int    msgSize;         // The size of this message
        double msgTimeStamp;    // The time that this message was generated (TRAFX_REQ)
        payload body;          // The data payload of this message
};
```



Appendix C

Configuration files for OMNET++ simulations

The settings of the OMNET++ simulations of TH(O)RP are controlled by three input files, namely 'layout.ini', 'parameters.ini' and 'omnetpp.ini'. The layout file sets the number of nodes, their positions and the settings of the communications channels. The parameters file set the parameters of TH(O)RP, for example message generation times and what fast recovery methods to enable.

The general configuration file, 'omnetpp.ini', sets general settings of the OMNET++ simulation environment and has default settings for the settings specified in the first two files. The general configuration file imports the first two files and they override certain settings of the simulation. If they are omitted from the general file, for instance they are not imported, then the simulation will execute with the default settings enabled. Take note that the general configuration file is not included here as users of this simulation would only modify the other two files to set up their simulations.

C.1 Layout file: layout.ini

```
#####
[Run 1]
description = "Compare against predicted latencies from latency models"
#####

[Parameters]
#####
#       Parameters for Netw/Appl layer on/off       #
#####

# Activate/de-activate individual nodes to aquire desired network.
#sim.m_host[1].net.active = false
#sim.m_host[2].net.active = false
#sim.m_host[3].net.active = false
#sim.m_host[4].net.active = false
#sim.m_host[5].net.active = false
#sim.m_host[6].net.active = false
sim.m_host[*].net.active = true
sim.c_host.net.active = true

sim.m_host[*].net.time_off = -1
sim.m_host[*].net.time_on = -1
```

```

sim.c_host.net.time_off = -1
sim.c_host.net.time_on = -1

# playgroundsize of the m_hosts (100 seperator, 100 on sides)
sim.playgroundSizeX = 500
sim.playgroundSizeY = 325

# number of m_hosts in the network
sim.numHosts = 6

# starting position for the m_hosts "-1" means random staring point
sim.m_host[0].mobility.x = 150;
sim.m_host[0].mobility.y = 175;

sim.m_host[1].mobility.x = 350;
sim.m_host[1].mobility.y = 175;

sim.m_host[2].mobility.x = 50;
sim.m_host[2].mobility.y = 275;

sim.m_host[3].mobility.x = 200;
sim.m_host[3].mobility.y = 275;

sim.m_host[4].mobility.x = 300;
sim.m_host[4].mobility.y = 275;

sim.m_host[5].mobility.x = 450;
sim.m_host[5].mobility.y = 275;

sim.m_host[*].mobility.x = -1;
sim.m_host[*].mobility.y = -1;

sim.c_host.mobility.x = 250;
sim.c_host.mobility.y = 75;

#####
[Run 2]
description = "Example simulation for sim_doc"
#####

[Parameters]
#####
#           Parameters for Netw/Appl layer on/off           #
#####
sim.m_host[*].net.time_off = -1
sim.m_host[*].net.time_on = -1
sim.c_host.net.time_off = -1
sim.c_host.net.time_on = -1

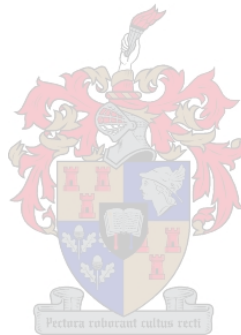
# playgroundsize of the m_hosts (100 seperator, 100 on sides)
sim.playgroundSizeX = 500
sim.playgroundSizeY = 500

# number of m_hosts in the network
sim.numHosts = 10

# starting position for the m_hosts "-1" means random staring point
sim.m_host[0].mobility.x = 150;
sim.m_host[0].mobility.y = 50;

sim.m_host[1].mobility.x = 250;
sim.m_host[1].mobility.y = 75;

```




```

sim.m_host[2].mobility.x = 350;
sim.m_host[2].mobility.y = 150;

sim.m_host[3].mobility.x = 450;
sim.m_host[3].mobility.y = 200;

sim.m_host[4].mobility.x = 450;
sim.m_host[4].mobility.y = 300;

sim.m_host[5].mobility.x = 300;
sim.m_host[5].mobility.y = 350;

sim.m_host[6].mobility.x = 150;
sim.m_host[6].mobility.y = 350;

sim.m_host[7].mobility.x = 100;
sim.m_host[7].mobility.y = 200;

sim.m_host[8].mobility.x = 50;
sim.m_host[8].mobility.y = 450;

sim.m_host[9].mobility.x = 200;
sim.m_host[9].mobility.y = 450;

sim.m_host[*].mobility.x = -1;
sim.m_host[*].mobility.y = -1;

sim.c_host.mobility.x = 250;
sim.c_host.mobility.y = 250;

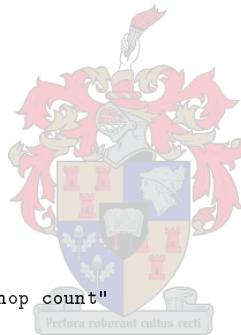
```

```
#####
```

```
[Run 3]
```

```
description = "Scenario to test effect of hop count"
```

```
#####
```



```
[Parameters]
```

```
#####
# Parameters for Netw/Appl layer on/off #
#####
```

```

sim.m_host[*].net.time_off = -1
sim.m_host[*].net.time_on = -1
sim.c_host.net.time_off = -1
sim.c_host.net.time_on = -1

```

```
# playgroundsize of the m_hosts (100 seperator, 100 on sides)
```

```

sim.playgroundSizeX = 1000
sim.playgroundSizeY = 1000

```

```
# number of m_hosts in the network
```

```
sim.numHosts = 20
```

```
# starting position for the m_hosts "-1" means random staring point
```

```

sim.m_host[0].mobility.x = 400;
sim.m_host[0].mobility.y = 300;

```

```

sim.m_host[1].mobility.x = 500;
sim.m_host[1].mobility.y = 400;

```

```

sim.m_host[2].mobility.x = 400;
sim.m_host[2].mobility.y = 500;

```

```

sim.m_host[3].mobility.x = 300;
sim.m_host[3].mobility.y = 500;

sim.m_host[4].mobility.x = 200;
sim.m_host[4].mobility.y = 400;

sim.m_host[5].mobility.x = 100;
sim.m_host[5].mobility.y = 500;

sim.m_host[6].mobility.x = 100;
sim.m_host[6].mobility.y = 600;

sim.m_host[7].mobility.x = 200;
sim.m_host[7].mobility.y = 700;

sim.m_host[8].mobility.x = 300;
sim.m_host[8].mobility.y = 800;

sim.m_host[9].mobility.x = 400;
sim.m_host[9].mobility.y = 800;

sim.m_host[10].mobility.x = 500;
sim.m_host[10].mobility.y = 800;

sim.m_host[11].mobility.x = 600;
sim.m_host[11].mobility.y = 700;

sim.m_host[12].mobility.x = 700;
sim.m_host[12].mobility.y = 600;

sim.m_host[13].mobility.x = 800;
sim.m_host[13].mobility.y = 500;

sim.m_host[14].mobility.x = 800;
sim.m_host[14].mobility.y = 400;

sim.m_host[15].mobility.x = 700;
sim.m_host[15].mobility.y = 300;

sim.m_host[16].mobility.x = 600;
sim.m_host[16].mobility.y = 200;

sim.m_host[17].mobility.x = 600;
sim.m_host[17].mobility.y = 100;

sim.m_host[18].mobility.x = 700;
sim.m_host[18].mobility.y = 50;

sim.m_host[19].mobility.x = 800;
sim.m_host[19].mobility.y = 50;

sim.m_host[*].mobility.x = -1;
sim.m_host[*].mobility.y = -1;

sim.c_host.mobility.x = 300;
sim.c_host.mobility.y = 200;

```



```

#####
[Run 4]
description = "General matrix simulation with standard settings"
#####
[Parameters]
#####

```

```

# Parameters for Netw/Appl layer on/off (-1 unused) #
#####
sim.m_host[*].active = true
sim.c_host.net.active = true
sim.m_host[*].net.time_off = -1
sim.m_host[*].net.time_on = -1
sim.c_host.net.time_off = -1
sim.c_host.net.time_on = -1

# playgroundsize of the m_hosts (160 seperator, 100 on sides)
sim.playgroundSizeX = 1000
sim.playgroundSizeY = 1000

# number of m_hosts in the network
sim.numHosts = 36

# starting position for the m_hosts "-1" means random staring point
# Row 1 #
sim.m_host[0].mobility.x = 100
sim.m_host[0].mobility.y = 100

sim.m_host[1].mobility.x = 260
sim.m_host[1].mobility.y = 100

sim.m_host[2].mobility.x = 420
sim.m_host[2].mobility.y = 100

sim.m_host[3].mobility.x = 580
sim.m_host[3].mobility.y = 100

sim.m_host[4].mobility.x = 740
sim.m_host[4].mobility.y = 100

sim.m_host[5].mobility.x = 900
sim.m_host[5].mobility.y = 100
# Row 1 #

# Row 2 #
sim.m_host[6].mobility.x = 100
sim.m_host[6].mobility.y = 260

sim.m_host[7].mobility.x = 260
sim.m_host[7].mobility.y = 260

sim.m_host[8].mobility.x = 420
sim.m_host[8].mobility.y = 260

sim.m_host[9].mobility.x = 580
sim.m_host[9].mobility.y = 260

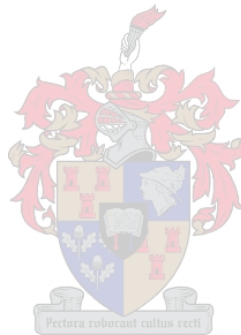
sim.m_host[10].mobility.x = 740
sim.m_host[10].mobility.y = 260

sim.m_host[11].mobility.x = 900
sim.m_host[11].mobility.y = 260
# Row 2 #

# Row 3 #
sim.m_host[12].mobility.x = 100
sim.m_host[12].mobility.y = 420

sim.m_host[13].mobility.x = 260
sim.m_host[13].mobility.y = 420

```



```
sim.m_host[14].mobility.x = 420
sim.m_host[14].mobility.y = 420

sim.m_host[15].mobility.x = 580
sim.m_host[15].mobility.y = 420

sim.m_host[16].mobility.x = 740
sim.m_host[16].mobility.y = 420

sim.m_host[17].mobility.x = 900
sim.m_host[17].mobility.y = 420
# Row 3 #

# Row 4 #
sim.m_host[18].mobility.x = 100
sim.m_host[18].mobility.y = 580

sim.m_host[19].mobility.x = 260
sim.m_host[19].mobility.y = 580

sim.m_host[20].mobility.x = 420
sim.m_host[20].mobility.y = 580

sim.m_host[21].mobility.x = 580
sim.m_host[21].mobility.y = 580

sim.m_host[22].mobility.x = 740
sim.m_host[22].mobility.y = 580

sim.m_host[23].mobility.x = 900
sim.m_host[23].mobility.y = 580
# Row 4 #

# Row 5 #
sim.m_host[24].mobility.x = 100
sim.m_host[24].mobility.y = 740

sim.m_host[25].mobility.x = 260
sim.m_host[25].mobility.y = 740

sim.m_host[26].mobility.x = 420
sim.m_host[26].mobility.y = 740

sim.m_host[27].mobility.x = 580
sim.m_host[27].mobility.y = 740

sim.m_host[28].mobility.x = 740
sim.m_host[28].mobility.y = 740

sim.m_host[29].mobility.x = 900
sim.m_host[29].mobility.y = 740
# Row 5 #

# Row 6 #
sim.m_host[30].mobility.x = 100
sim.m_host[30].mobility.y = 900

sim.m_host[31].mobility.x = 260
sim.m_host[31].mobility.y = 900

sim.m_host[32].mobility.x = 420
sim.m_host[32].mobility.y = 900
```



```
sim.m_host[33].mobility.x = 580
sim.m_host[33].mobility.y = 900
```

```
sim.m_host[34].mobility.x = 740
sim.m_host[34].mobility.y = 900
```

```
sim.m_host[35].mobility.x = 900
sim.m_host[35].mobility.y = 900
# Row 6 #
```

```
sim.c_host.mobility.x = 500;
sim.c_host.mobility.y = 500;
```

```
#####
```

```
[Run 5]
```

```
description = "Random node placement"
```

```
#####
```

```
[Parameters]
```

```
#####
```

```
# Parameters for Netw/Appl layer on/off #
```

```
#####
```

```
sim.m_host[*].net.time_off = -11
```

```
sim.m_host[*].net.time_on = -1
```

```
sim.c_host.net.time_off = -1
```

```
sim.c_host.net.time_on = -1
```

```
# playgroundsize of the m_hosts (100 seperator, 100 on sides)
```

```
sim.playgroundSizeX = 1000
```

```
sim.playgroundSizeY = 1000
```

```
# number of m_hosts in the network
```

```
sim.numHosts = 100
```

```
# starting position for the m_hosts "-1" means random staring point
```

```
sim.m_host[*].mobility.x = -1;
```

```
sim.m_host[*].mobility.y = -1;
```

```
sim.c_host.mobility.x = 300;
```

```
sim.c_host.mobility.y = 200;
```

```
#####
```

```
[Run 6]
```

```
description = "Random node placement"
```

```
#####
```

```
[Parameters]
```

```
#####
```

```
# Parameters for Netw/Appl layer on/off #
```

```
#####
```

```
sim.m_host[*].net.time_off = -1
```

```
sim.m_host[*].net.time_on = -1
```

```
sim.c_host.net.time_off = -1
```

```
sim.c_host.net.time_on = -1
```

```
# playgroundsize of the m_hosts (100 seperator, 100 on sides)
```

```
sim.playgroundSizeX = 1600
```

```
sim.playgroundSizeY = 1600
```

```
# number of m_hosts in the network
```



```

sim.numHosts = 225

# starting position for the m_hosts "-1" means random starting point
sim.m_host[*].mobility.x = -1;
sim.m_host[*].mobility.y = -1;

sim.c_host.mobility.x = 800;
sim.c_host.mobility.y = 800;

#####
#       Parameters for the ChannelControl                               #
#####

# carrier frequency in hertz
sim.channelcontrol.carrierFrequency = 2.4e+9
# max transmission power [mW]
sim.channelcontrol.pMax = 2.0
# signal attenuation threshold [dBm]
sim.channelcontrol.sat = -82
# path loss coefficient alpha
sim.channelcontrol.alpha = 2.0

#####
#       Parameters for the MAC Layer                                   #
#####

# debug switch for the MAC layer
sim.m_host[*].nic.mac.debug = false
sim.c_host.nic.mac.debug = false

sim.m_host[*].nic.mac.headerLength=272
sim.m_host[*].nic.mac.maxQueueSize=14
sim.m_host[*].nic.mac.bitrate=2E+6; in bits/second
sim.m_host[*].nic.mac.rtsCts=false
sim.m_host[*].nic.mac.broadcastBackoff=31

sim.c_host.nic.mac.headerLength=272
sim.c_host.nic.mac.maxQueueSize=14
sim.c_host.nic.mac.bitrate=2E+6; in bits/second
sim.c_host.nic.mac.rtsCts=false
sim.c_host.nic.mac.broadcastBackoff=31

# MAC message header length
sim.m_host[*].mac.headerLength=24
sim.c_host.mac.headerLength=24

#####
#       Parameters for the Decider                                     #
#####

# core MF debug switch for the decider
sim.m_host[*].nic.decider.debug = false
sim.c_host.nic.decider.debug = false

sim.m_host[*].nic.decider.snirThreshold=4; in dB
sim.m_host[*].nic.decider.bitrate=2E+6; in bits/second

sim.c_host.nic.decider.snirThreshold=4; in dB
sim.c_host.nic.decider.bitrate=2E+6; in bits/second

#####

```



```

#           Parameters for the SnrEval                                     #
#####

# debug switch for the snrEval
sim.m_host[*].nic.snrEval.coreDebug = false
sim.c_host.nic.snrEval.coreDebug = false

# debug switch
sim.m_host[*].nic.snrEval.debug = true
sim.c_host.nic.snrEval.debug = true

# AirFrame header length
sim.m_host[*].nic.snrEval.headerLength=192
sim.c_host.nic.snrEval.headerLength=192

# bitrate [bits/s]
sim.m_host[*].nic.snrEval.bitrate=2E+6; in bits/second
sim.c_host.nic.snrEval.bitrate=2E+6; in bits/second

# transmission power [mW]
sim.m_host[*].nic.snrEval.transmitterPower=2; [mW]
sim.c_host.nic.snrEval.transmitterPower=2; [mW]

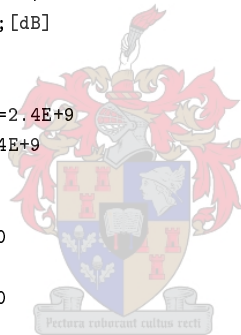
# SNR threshold
sim.m_host[*].nic.snrEval.snrThresholdLevel=3;[dB]
sim.c_host.nic.snrEval.snrThresholdLevel=3;[dB]

# Carrier frequency
sim.m_host[*].nic.snrEval.carrierFrequency=2.4E+9
sim.c_host.nic.snrEval.carrierFrequency=2.4E+9

# Other TX/RX antenna parameters
sim.m_host[*].nic.snrEval.thermalNoise=-110
sim.m_host[*].nic.snrEval.sensitivity=-85
sim.m_host[*].nic.snrEval.pathLossAlpha=2.0

sim.c_host.nic.snrEval.thermalNoise=-110
sim.c_host.nic.snrEval.sensitivity=-85
sim.c_host.nic.snrEval.pathLossAlpha=2.2

```



```

#####
#           Channel paramters (default = 0)                             #
#####
# [NOTE: Take care not to enforce double drop rate! (unless intended)]
sim.m_host[*].net.in_channel_drop_rate = 5;
sim.c_host.net.in_channel_drop_rate = 5;
sim.m_host[*].net.out_channel_drop_rate = 5;
sim.c_host.net.out_channel_drop_rate = 5;

```

C.2 Parameter file: parameters.ini

```

#####
#           Set TH(0)RP parameters (use -1 for default values)         #
#####
[Parameters]

# Simulation output msgs toggles (visual aids)
sim.m_host[*].net.data_on = true
sim.m_host[*].net.ack_on = false
sim.m_host[*].net.route_on = false
sim.m_host[*].net.mid_hna_on = false

```

```

sim.m_host[*].net.hello_on = false

sim.c_host.net.data_on = true
sim.c_host.net.ack_on = false
sim.c_host.net.route_on = false
sim.c_host.net.mid_hna_on = false
sim.c_host.net.hello_on = false

# Appl layer parameter
sim.m_host[*].appl.alive = -1
sim.c_host.appl.alive = -1

# TH(0)RP parameters
# Message generation intervals
sim.m_host[*].net.hello_interval = -1
sim.m_host[*].net.parent_sel_interval = -1
sim.m_host[*].net.route_update_interval = -1
sim.m_host[*].net.mid_interval = -1
sim.m_host[*].net.hna_interval = -1

sim.c_host.net.hello_interval = -1
sim.c_host.net.parent_sel_interval = -1
sim.c_host.net.route_update_interval = -1
sim.c_host.net.mid_interval = -1
sim.c_host.net.hna_interval = -1

# Hold times
sim.m_host[*].net.dup_hold_time = 10      # reduced from 30s due to absence of multiple routes
sim.m_host[*].net.retry_time = 0.1875    # 1.5 * max_jitter

sim.c_host.net.dup_hold_time = 10        # reduced from 30s due to absence of multiple routes
sim.c_host.net.retry_time = 0.1875      # 1.5 * max_jitter

# Maximum forwarding jitter and retries
sim.m_host[*].net.max_jitter = -1
sim.m_host[*].net.num_retries = -1

sim.c_host.net.max_jitter = -1
sim.c_host.net.num_retries = -1

#####
# Fast recovery/hierarchy construction methods
#####
# Fast recovery (RERR)
sim.m_host[*].net.rerr_on = true
sim.c_host.net.rerr_on = true

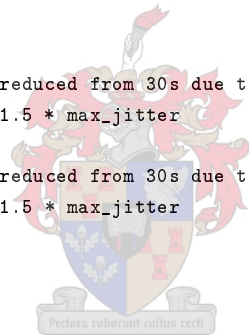
# Fast recovery (increased pselect interval)
sim.m_host[*].net.inc_psel_int = true
sim.c_host.net.inc_psel_int = true

# AHSPP
sim.m_host[*].net.ahspp_on = true
sim.c_host.net.ahspp_on = true

# Adoption announcements
sim.m_host[*].net.adopt_ann = true
sim.c_host.net.adopt_ann = true

# Cummalitive rreq
sim.m_host[*].net.crreq_on = true
sim.c_host.net.crreq_on = true

```



Appendix D

Matlab code for latency calculator

A Matlab script was created to calculate the latencies of the various latency models developed in this thesis. The 'lat.m' script implements the various models from Chapter 5, and can provide output for a specific scenario specified by its input parameters. A user can specify the device bandwidth, the average forwarding jitter, the channel error rate, the latency model and direction of communications and the technology used. For example, to calculate the latency for GPRS at 80kb/s, with a 0.5s forwarding jitter, 10% channel drop rate, and on the N:N:1 model in the upward direction enter the following in Matlab (with the working directory set to the location of the scripts):

```
lat(80e3,0.5,10,3,0,4)
```

For a list of the available input parameters reference the 'lat.m' script.

Because we investigated a wide range of technologies and bandwidths over a multitude of models, it became increasingly timely to calculate the latencies for each situation. To remedy this problem a script, 'tech_lat.m' was written that executed 'lat.m' for all the different models and technologies and compiled the results together. The 'tech_lat.m' script can also output the collected results in a TeX friendly output, which tabularises the gathered information for each technology and model. These tables can be seen in Table 2.4 and 2.5. The TeX output from 'tech_lat.m' is stored in a file named 'lat_output.txt' and can be located in the same directory as where 'tech_lat.m' was executed from. In order to calculate all the latencies, with a 0.125s maximum forwarding jitter and 15% channel error rate and write the output in a TeX friendly format execute the following in Matlab:

```
tech_lat(2,0.125,10)
```

D.1 Individual latency calculator: 'lat.m'

```
function [T, P, Rtt] = latency(bw, fwd_jit, err_rate, model, dir, iface)
% Calculate the latency for a given bandwidth, interface type, model and
% direction.
%-----
% Author - DW. Morrison
% Date - 21/07/06
% Project - M.Sce(Eng)
%-----
% Input
% bw      - Device bandwidth in bit/s
```

```

% fwd_jit - The absolute maximum forwarding jitter
% err_rate - Communications error rate (specify 10 for a 10% error rate)
% iface - Network interface type
% model - Queueing model
% dir - Direction of traffic
%-----
% Output
% T - Wait time in milli seconds
% stab - From traffic intensity ( < 1 is stable, >= 1 is unstable )
%-----
% Model values
% 0 - 1:1 (Default)
% 1 - N:1
% 2 - 1:1:1
% 3 - N:N:1
% 4 - 1:1 with retransmissions
%-----
% Direction values
% 0 - Up (Default)
% 1 - Down
%-----
% Interface values
% 0 - eth (ethernet)
% 1 - OC (optical carrier)
% 2 - 802.11 (Wi-fi) (Default)
% 3 - 802.16 (WiMAX)
% 4 - GPRS
% 5 - 3G
% 6 - BPL (PLC)
%-----
format long

% Set default values
if (nargin < 1)
    bw = 2e6; fwd_jit = 0; err_rate = 0; model = 0; dir = 0; iface = 2;
end;
if (nargin < 2)
    fwd_jit = 0; err_rate = 0; model = 0; dir = 0; iface = 2;
end;
if (nargin < 3)
    err_rate = 0; model = 0; dir = 0; iface = 2;
end;
if (nargin < 4)
    model = 0; dir = 0; iface = 2;
end;
if (nargin < 5)
    dir = 0; iface = 2;
end;
if (nargin < 6)
    iface = 2;
end;

% Calculate the error rate probability
if( err_rate == 0 )
    p_err = 0;
else
    p_err = err_rate/100;
end;

% If bw is zero, don't calculate latency
Rtt = 0; % Zero RTT
T = 0; % Zero latency
P = 1; % Unstable

```



```

if(bw > 0)
    % Constants
    %-----
    % Generation rates
    Hello_gen = 1/2;   HNA_gen = 1/5;   MID_gen = 1/5;
    Route_gen = 1/6;   ACK_route_gen = 1/6;
    Data_gen = 1;      ACK_data_gen = 1;
    % THRP params
    % Maximum BCAST size (Ethernet MTU - IPv4_UDP header)
    THRP_MAX = (1500-28)*8;
    % Header size
    THRP_P_head = 32;   THRP_M_head = 96;   Hello_head = 32;
    MID_head = 0;      HNA_head = 0;      Route_head = 32;
    Data_head = 128;   ACK_head = 64;
    % Payload size
    Hello_body = 384;   MID_body = 64;      HNA_body = 64;
    Route_body = 128;   Data_body = 800;

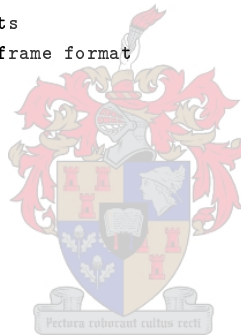
    % Socket headers
    IPv4_UDP_head = 224;
    % MAC headers
    ETH_head = 72;
    OC_head = 72;      %NOTE: Still need frame format
    WiFi_head = 272;
    WiMAX_head = 51;
    GPRS_head = 504;   %NOTE: Used 12 slots
    ThreeG_head = 504; %NOTE: Still need frame format
    %-----

    % Initialize variables
    %-----
    % Arrival rate
    A1 = 0; A2 = 0; A3 = 0;
    % Service rate
    S1 = 0; S2 = 0; S3 = 0;
    % Stability
    P1 = 0; P2 = 0; P3 = 0;
    % Wait time (latency)
    T = 0; T1 = 0; T2 = 0;
    % Data sizes
    Dt1 = 0; Dt2 = 0; Dup = 0; Dh = 0; Ddown = 0; Dco = 0; Dthrph = 0;
    %-----

    % Calculate interface header
    %-----
    if( iface==0 )
        MAC_head = ETH_head;
    elseif( iface==1 )
        MAC_head = OC_head;
    elseif( iface==2 )
        MAC_head = WiFi_head;
    elseif( iface==3 )
        MAC_head = WiMAX_head;
    elseif( iface==4 )
        MAC_head = GPRS_head;
    elseif( iface==5 )
        MAC_head = ThreeG_head;
    elseif( iface==6 )
        MAC_head = ETH_head;
    end;

    %-----
    % Calculate data sizes

```



```

%-----
if(dir == 0)
    % Up
    Dup = Hello_gen*Hello_body + MID_gen*MID_body + HNA_gen*HNA_body + ...
        + Route_gen*Route_body + Data_gen*Data_body;
    Dh = Hello_gen*(Hello_head+THRP_M_head) + MID_gen*(MID_head+THRP_M_head) + ...
        + HNA_gen*(HNA_head+THRP_M_head) + Route_gen*(Route_head+ACK_head+2*THRP_M_head) + ...
        + Data_gen*(Data_head+ACK_head+2*THRP_M_head);
    if(model <= 1 | model == 4)
        % Add THRP headers
        Dt1 = Dup + Dh;
        Dt1 = Dt1 + ceil((Dt1)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
        if( model == 4 )
            % Adjust generated data to include retransmissions
            Dt1 = (1 + p_err)*Dt1;
        end;
    elseif(model == 2)
        Dt1 = Dup + Dh;
        Dco = MID_gen*(MID_body+MID_head+THRP_M_head) + HNA_gen*(HNA_body+HNA_head+THRP_M_head) + ...
            + Route_gen*(Route_body+Route_head+ACK_head+2*THRP_M_head) + ...
            + Data_gen*(Data_body+Data_head+ACK_head+2*THRP_M_head);
        Dt2 = Dt1 + Dco;
        % Add THRP headers
        Dt1 = Dt1 + ceil((Dt1)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
        Dt2 = Dt2 + ceil((Dt2)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
    elseif(model == 3)
        Dt1 = Dup + Dh;
        Dco = MID_gen*(MID_body+MID_head+THRP_M_head) + HNA_gen*(HNA_body+HNA_head+THRP_M_head) + ...
            + Route_gen*(Route_body+Route_head+ACK_head+2*THRP_M_head) + ...
            + Data_gen*(Data_body+Data_head+ACK_head+2*THRP_M_head);
        Dt2 = 2*(Dt1 + Dco);
        % Add THRP headers
        Dt1 = Dt1 + ceil((Dt1)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
        Dt2 = Dt2 + ceil((Dt2)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
    end;
else
    % Down
    Ddown = Hello_gen*Hello_body + MID_gen*MID_body + HNA_gen*HNA_body + ...
        + Route_gen*Route_body + Data_gen*Data_body;
    Dh = Hello_gen*(Hello_head+THRP_M_head) + MID_gen*(MID_head+THRP_M_head) + ...
        + HNA_gen*(HNA_head+THRP_M_head) + Route_gen*(Route_head+ACK_head+2*THRP_M_head) + ...
        + Data_gen*(Data_head+ACK_head+2*THRP_M_head);
    if(model <= 1 | model == 4)
        % Add THRP headers
        Dt1 = Ddown + Dh;
        Dt1 = Dt1 + ceil((Dt1)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
        if( model == 4 )
            % Adjust generated data to include retransmissions
            Dt1 = (1 + p_err)*Dt1;
        end;
    else
        Dt1 = Ddown + Dh;
        Dco = MID_gen*(MID_body+MID_head+THRP_M_head) + HNA_gen*(HNA_body+HNA_head+THRP_M_head) + ...
            + Route_gen*(Route_body+Route_head+ACK_head+2*THRP_M_head) + ...
            + Data_gen*(Data_body+Data_head+ACK_head+2*THRP_M_head);
        Dt2 = Dt1 + Dco;
        % Add THRP headers
        Dt1 = Dt1 + ceil((Dt1)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
        Dt2 = Dt2 + ceil((Dt2)/(THRP_MAX - THRP_P_head))*(THRP_P_head + IPv4_UDP_head + MAC_head);
    end;
end;

% Calculate arrival rates

```

```

%-----
if(dir == 0)
    % Up
    if(model == 0 | model == 4)
        A1 = Hello_gen + MID_gen + HNA_gen + Route_gen + ACK_route_gen + Data_gen + ACK_data_gen;
    elseif(model == 1)
        A1 = Hello_gen + MID_gen + HNA_gen + Route_gen + ACK_route_gen + Data_gen + ACK_data_gen;
        A1 = 2*A1;
    elseif(model == 2)
        A1 = Hello_gen + MID_gen + HNA_gen + Route_gen + ACK_route_gen + Data_gen + ACK_data_gen;
    elseif(model == 3)
        A1 = Hello_gen + MID_gen + HNA_gen + Route_gen + ACK_route_gen + Data_gen + ACK_data_gen;
        A1 = 2*A1; A2 = A1;
        A3 = A1+A2;
    end;
else
    % Down
    A1 = Hello_gen + MID_gen + HNA_gen + Route_gen + ACK_route_gen + Data_gen + ACK_data_gen;
end;

% Calculate service rates
%-----
if(dir == 0)
    % Up
    if(model == 0 | model == 1 | model == 4)
        S1 = bw/Dt1; S2 = 0; S3 = 0;
    elseif(model == 2)
        S1 = bw/Dt1; S2 = bw/Dt2; S3 = 0;
    elseif(model == 3)
        S1 = bw/Dt1; S2 = bw/Dt1; S3 = bw/Dt2;
    end;
else
    % Down
    if(model == 0 | model == 4)
        S1 = bw/Dt1; S2 = 999999; S3 = 999999;
    elseif(model == 1)
        S1 = bw/Dt1; S2 = bw/Dt1; S3 = 999999;
    elseif(model == 2)
        S1 = bw/Dt1; S2 = bw/Dt2; S3 = 999999;
    elseif(model == 3)
        S1 = bw/Dt1; S2 = bw/Dt1; S3 = bw/Dt2;
    end;
end;

% Calculate latency
%-----
if(dir == 0)
    % Up
    if(model == 0 | model == 4)
        T = 1/(S1-A1);
    elseif(model == 1)
        T = 1/(S1-A1);
    elseif(model == 2)
        T1 = 1/(S1-A1); T2 = 1/(S2-A1); T = T1 + T2;
    elseif(model == 3)
        T1 = 1/(S1-A1); T2 = 1/(S3-A3); T = T1 + T2;
    end;
else
    % Down
    if(model == 0 | model == 4)
        T = 1/(S1-A1);
    elseif(model == 1)
        T = 1/(S1-A1);

```

```

elseif(model == 2)
    T1 = 1/(S1-A1); T2 = 1/(S2-A1); T = T1 + T2;
elseif(model == 3)
    T1 = 1/(S1-A1); T2 = 1/(S3-A1); T = T1 + T2;
end;
end;

% Calculate stability
%-----
P = 0;
if( (T <= 0) | (A1/S1 >= 1) | (A2/S2 >= 1) | (A3/S3 >= 1) )
    P = 1;
end;

% Add the effects of the forwarding jitter to latency
N = 0;
if( model <= 1 | model == 4 )
    N = 1;
elseif( model == 2 | model == 3 )
    N = 2;
end;
% If queue is stable add forwarding jitter
if( P == 0 )
    T = T + N*fwd_jit;
end;
end;

Rtt = 2*T;

```

D.2 Technology latency calculator: 'tech_lat.m'

```

function [T1, T2, T3, T4, T5, T6, T7, T8, R1, R2, R3, R4, R5, R6, R7, R8] = tech_lat(mode, max_fwd_jitter, error_rate)
% Calculate the latencies for all models with input bandwidths.
% mode          : Can specify normal output to command line or write output to
%                file in TeX friendly format.
% max_fwd_jitter: The maximum forwarding jitter of TH(0)RP nodes
% error_rate    : The channel error rate (specify 10 for a 10% error rate)

% Set default values
if (nargin < 1)
    mode = 0;
end;
if (mode>=1)
    % Open output file and blank it. (normal output)
    f = fopen('lat_output.txt', 'w');
    if (mode==2)
        % TeX output mode

        % Constants
        Tech_BW_S = {'40Gb/s' '155Mb/s' '40Gb/s' '155Mb/s';
                    '2.7Mb/s' '256kb/s' '2.7Mb/s' '256kb/s';
                    '11Mb/s' '1Mb/s' '11Mb/s' '1Mb/s';
                    '54Mb/s' '2Mb/s' '54Mb/s' '2Mb/s';
                    '70Mb/s' '2Mb/s' '70Mb/s' '2Mb/s';
                    '80kb/s' '28.8kb/s' '80kb/s' '57.6kb/s';
                    '236.8kb/s' '170kb/s' '236.8kb/s' '170kb/s';
                    '2Mb/s' '384kb/s' '2Mb/s' '384kb/s';
                    '10.8Mb/s' '1.8Mb/s' '10.8Mb/s' '1.8Mb/s'};

        % Technology names
        Names = str2mat('OC', 'BPL', 'Wi-Fi (11Mbps)', 'Wi-Fi (54Mbps)', ...

```

```

        'WiMAX','GPRS','EDGE','3G','HSDPA');
    end;
end;

% Constants
% Specify the technology bandwidths
Tech_BW = [40e9 155e6 40e9 155e6;
           2.7e6 256e3 2.7e6 256e3;
           11e6 1e6 11e6 1e6;
           54e6 2e6 54e6 2e6;
           70e6 2e6 70e6 2e6;
           80e3 28.8e3 80e3 57.6e3;
           236.8e3 170e3 236.8e3 170e3;
           2e6 384e3 2e6 384e3;
           10.8e6 1.8e6 10.8e6 1.8e6];

% Caclulate the average forwarding jitter from the absolute maximum
fwd_jitter = max_fwd_jitter/2;

% Initialise variable
% Latencies
Tm = [zeros(10,2)];
T1 = Tm; T2 = Tm; T3 = Tm; T4 = Tm; T5 = Tm; T6 = Tm; T7 = Tm; T8 = Tm; T9 = Tm;

% Round trip times
Rm = [zeros(10,2)];
R1 = Rm; R2 = Rm; R3 = Rm; R4 = Rm; R5 = Rm; R6 = Rm; R7 = Rm; R8 = Rm; R9 = Rm;

% Stabilities
Pm = [zeros(10,2)];
P1 = Pm; P2 = Pm; P3 = Pm; P4 = Pm; P5 = Pm; P6 = Pm; P7 = Pm; P8 = Pm; P9 = Pm;

for j=1:9,
    % Tech
    k = 1;
    for i=1:10,
        % Model
        if(mod(i-1,2) == 0)
            % Up
            [Tm(i,1), Pm(i,1), Rm(i,1)] = lat(Tech_BW(j,1),fwd_jitter,error_rate,i-k,0,getIface(j));
            [Tm(i,2), Pm(i,2), Rm(i,2)] = lat(Tech_BW(j,2),fwd_jitter,error_rate,i-k,0,getIface(j));
            k = k + 1;
        else
            % Down
            [Tm(i,1), Pm(i,1), Rm(i,1)] = lat(Tech_BW(j,3),fwd_jitter,error_rate,i-k,1,getIface(j));
            [Tm(i,2), Pm(i,2), Rm(i,2)] = lat(Tech_BW(j,4),fwd_jitter,error_rate,i-k,1,getIface(j));
        end
    end
end
if(j==1)
    T1 = Tm; P1 = Pm; R1 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(1,:), Tech_BW_S(1,:), T1, P1, R1);
    else
        writeResults(f, mode, Names(1,:), Tech_BW_S(1,:), T1(:,1), P1(:,1), R1(:,1));
    end;
elseif(j==2)
    T2 = Tm; P2 = Pm; R2 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(2,:), Tech_BW_S(2,:), T2, P2, R2);
    else
        writeResults(f, mode, Names(2,:), Tech_BW_S(2,:), T2(:,1), P2(:,1), R2(:,1));
    end;
end;

```

```

elseif(j==3)
    T3 = Tm; P3 = Pm; R3 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(3,:), Tech_BW_S(3,:), T3, P3, R3);
    else
        writeResults(f, mode, Names(3,:), Tech_BW_S(3,:), T3(:,1), P3(:,1), R3(:,1));
    end;
elseif(j==4)
    T4 = Tm; P4 = Pm; R4 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(4,:), Tech_BW_S(4,:), T4, P4, R4);
    else
        writeResults(f, mode, Names(4,:), Tech_BW_S(4,:), T4(:,1), P4(:,1), R4(:,1));
    end;
elseif(j==5)
    T5 = Tm; P5 = Pm; R5 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(5,:), Tech_BW_S(5,:), T5, P5, R5);
    else
        writeResults(f, mode, Names(5,:), Tech_BW_S(5,:), T5(:,1), P5(:,1), R5(:,1));
    end;
elseif(j==6)
    T6 = Tm; P6 = Pm; R6 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(6,:), Tech_BW_S(6,:), T6, P6, R6);
    else
        writeResults(f, mode, Names(6,:), Tech_BW_S(6,:), T6(:,1), P6(:,1), R6(:,1));
    end;
elseif(j==7)
    T7 = Tm; P7 = Pm; R7 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(7,:), Tech_BW_S(7,:), T7, P7, R7);
    else
        writeResults(f, mode, Names(7,:), Tech_BW_S(7,:), T7(:,1), P7(:,1), R7(:,1));
    end;
elseif(j==8)
    T8 = Tm; P8 = Pm; R8 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(8,:), Tech_BW_S(8,:), T8, P8, R8);
    else
        writeResults(f, mode, Names(8,:), Tech_BW_S(8,:), T8(:,1), P8(:,1), R8(:,1));
    end;
elseif(j==9)
    T9 = Tm; P9 = Pm; R9 = Rm;
    if( Tech_BW(j,2) > 0 )
        writeResults(f, mode, Names(9,:), Tech_BW_S(9,:), T9, P9, R9);
    else
        writeResults(f, mode, Names(9,:), Tech_BW_S(9,:), T9(:,1), P9(:,1), R9(:,1));
    end;
end;
end;

% Close the output file
if (mode>=1)
    fclose(f);
end;

```


Appendix E

Proposal presented to the CoCT

Project Proposal: Traffic Controller Communication Investigation for Advanced Networking Solution

PROJECT DESCRIPTION

Traffic controller communication is vital to obtain successful traffic flow and to monitor the status of traffic at signalised intersections. The City of Cape Town has a need for a reliable, expandable, low latency, time verifiable network that can provide communication between the traffic controllers and a central computer.

The communication system needs to provide the following functionality:

- download/upload traffic flow algorithms to/from traffic controllers
- monitor traffic and controller status at signalised intersections

The requirements of the communications system are:

- reliable
- expandable
- low latency
- robust
- time verifiable (time stamped data) within tight parameters
- geocoded information

The goal of the project is to investigate different networking technologies that would satisfy the

requirements stated above. Technologies that will be investigated are conventional wired technologies and conventional telemetry radio (Wi-Fi, WiMAX and GPRS) along with an impact study of the extremely variable propagation delay characteristics in metropolitan areas. The investigation will focus on ad-hoc networks as a possible solution. A suitable technology will then be chosen and a suitable protocol developed. The system will then be simulated in software and a hardware prototype will subsequently be developed and tested.

Upon successful completion of the project a final report with results and recommendations will be prepared and presented to the City of Cape Town. The final report will make recommendations based on various criteria, including (but not limited to):

1. Technical Performance;
2. Cost;
3. Sustainability;
4. Expandability / Modularity;
5. Future of technology (How 'future proof' is it?);
6. Community benefits;
7. Maintenance Cost;
8. People and skills required to maintain and operate.

A weight will be allocated to each based on the City of Cape Town's priorities at the time of evaluation. From the recommendations a suitable solution can be chosen and a pilot project implemented. Continuous contact and consultation with the City of Cape Town will be retained throughout.

STATEMENT OF PROBLEM

Efficient traffic management is vital to the successful operation of transport in any metropolitan area. The

need for efficient management is further stressed by Cape Town's unique geography, with Table Mountain and the ocean limiting the space available for expansion. The 2010 soccer world cup will add to the congestion on Cape Town's roads and the success of the tournament will rely on the ability of players and spectators to reach the stadiums on-time and without incident. An effective public transport system is also required in the City of Cape Town and at present the infrastructure does not exist to implement such a system.

It is therefore crucial to implement a traffic management system that can monitor traffic and download traffic routing algorithms remotely to help alleviate the current situation. With a traffic management system public transport can be prioritised, thereby promoting the use of public transport in the City of Cape Town. Efficient transport systems will also promote the growth and prosperity of metropolitan areas through increased productivity of the work force (reduced rush hour traffic) with an obvious reduction in traffic accidents and will enable the City of Cape Town and South Africa to compete in the global community.

The current situation arose from the lack of infrastructure to provide such a communications system. Telkom supplied the original infrastructure and still maintains the existing infrastructure, but due to advances in technology at Telkom they can no longer expand that infrastructure. Thus the current infrastructure cannot be extended and this has a crippling impact on the transport development strategy of the City of Cape Town. The new technology offered by Telkom has high costs involved and this project aims to identify more cost effective solutions.

With that in mind and the ever increasing traffic on South Africa's roads it is imperative to have a communication system in place that can provide the means to achieve this traffic management, which includes providing real time information to operators and the public. Through a thorough study of different technologies, a tailor made solution can be designed and implemented which could be modified and expanded as required.

The use of a university student for the required research has significant cost benefits and benefits from the exceptional infrastructure offered by the university along with the expertise of senior staff members affiliated with the project. This will also result in the

training of the student for future employment whom can bring the expertise acquired during the project to the table. Such a person would be vital to the future success of the finished product.

ADDITIONAL PROJECT BENEFITS

The project aims to design a system that will be highly expandable. This will be accomplished by including redundancy in the systems design through a surplus of network resources. This surplus can - for example - be used to operate Variable Message Signs (VMS) and provide some form of community Internet access.

The project aims to promote iKapa Elihlumayo (The Growing Cape). This could possibly be achieved in the future by providing basic communications services to the community by using the surplus of resources in the proposed system. By contributing to efforts to provide a basic communications service to the community the project can enable the City of Cape Town to empower her people, investing in building social and human capital by providing basic education through communication. The project also aims to promote Batho Pele (people first) by improving the quality of life of the community through job creation and the promotion of public transport, improving the environment by reducing air and noise pollution. With effective transport management and public transport systems in place the City of Cape Town can promote economic development and tourism.

The system can also be employed to improve coordination between traffic management and emergency services. This can be achieved by employing the traffic management system to prioritise routes for ambulances and fire trucks. The system can be used to monitor and evaluate the effectiveness of the transport strategies by providing a real time representation of what is happening on the roads.

This project and the results obtained therein could be used to make the Cape 'A Home for All'.

PROJECT DETAIL

Goals

There are seven major goals for the project:

- **Goal 1** - Thorough understanding of the requirements for the communications system.

- **Goal 2** - Investigate the suitability of current technologies.
- **Goal 3** - Select the most applicable technology and adapt for use as required with emphasis on protocol development and possible ad-hoc networks.
- **Goal 4** - Develop and test a hardware prototype of the communication system.
- **Goal 5** - Produce the results and a solution to the Transport Directorate of the City of Cape Town.
- **Goal 6** - Produce a masters thesis for a MSc(Eng) degree at the University of Stellenbosch.
- **Goal 7** - Train the student for possible employment in this field.

Interest groups

There are two interest groups for this project:

- The first, and primary, interest group is the Transport Directorate of the City of Cape Town. The project is being conducted for use by the primary group. This group is represented in this project by Goals 1 through 5.
- The second interest group is the University of Stellenbosch, a student, Daniel Weich Morrison, and other members of the Electrical Engineering Faculty at the University of Stellenbosch. The project will be conducted by the secondary group in co-operation with the primary group. This group is represented in this project by Goals 6 and 7.

Methods

The primary methods for achieving the goals of the project will be:

- Close communication between both interest groups will ensure that the goals are met and that they satisfy both sides' needs.
- A thorough literature study and research into applicable technologies will be conducted and the best possible solution will be used. A mathematical model for the system will be derived from this study.
- The mathematical model will be simulated in software to verify the correct operation.
- A hardware prototype will be implemented to verify correct operation, and from the prototype a final product will be derived. The prototype

along with a model for the final product will be presented to the Transport Directorate of the City of Cape Town.

In addition, at the conclusion of the project a detailed report about the project will be compiled and handed to the Transport Directorate of the City of Cape Town to provide information about the outcome of the project. This report will form the basis to roll out the project and will provide information applicable to similar projects. A contract will be drafted between the City of Cape Town and the University of Stellenbosch over the intellectual information property.

Staff/Administration

The project will employ one full time and four part time staff members:

- Masters student (full time) - Responsible for research, simulation and testing. This person will also develop and test a hardware prototype and will compile the results for the project. The Masters student will be Daniel Weich Morrison (co-author of this proposal - CV attached).
- Study leader/Project head (part time) - Responsible for providing guidance and expertise to Daniel Weich Morrison during the course of the Project. The study leader will be Dr R Wolhuter (co-author of this proposal).
- Additional assistants (part time) - Responsible for providing additional assistance and expertise in their respective fields. The additional assistants will be Prof JG Lourens, (co-author of this proposal and Telecommunications Group leader at the University of Stellenbosch), and Dr G-J van Rooyen.
- Representative at the Transport Directorate of the City of Cape Town (part time) - Responsible for acting as contact person at the Transport Directorate and for providing guidance during project. Will guide direction of project according to the Transport Directorate' requirements and will evaluate success of project relative to the Transport Directorate' needs. This role will be filled by Mr Francois Nell, who will conduct the management aspects, and Mr Craig Graham, who will conduct the technical aspects.

Available resources

- **Computer facilities** - Provided by Stellenbosch University.

- **Electronic laboratories** - Provided by Stellenbosch University.
- **Development software** - Provided by Stellenbosch University and Dr R Wolluter.

Appendix A - Time line

Required resources

- **Personnel** - One full time and four part time (personnel will be constituted from members interested in this project).
- **Facilities** - None
- **Equipment**
 - Traffic controller (Sintell/Siemens)
 - Development software packages for traffic controller.
 - Detailed digital maps and digital elevation models of the City of Cape Town.
- **Hardware** - ISM band hardware for development

Evaluation plan

Project evaluation will be the responsibility of all parties closely associated with the project. The results obtained will be evaluated and an applicable solution will be derived from the results. The success and feasibility of the suggested solution will be verified by means of mathematical models, software simulation and hardware prototypes. It is also proposed that a pilot installation be done in a limited area, to verify the performance in the most realistic way possible.

