Department of Electrical & Computer Engineering

Architecture and Performance of Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP)

Steven Martijn Van Der Werf

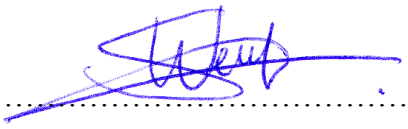
This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University of Technology

November 2010

DECLARATION

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgment has been made.

This thesis contains no material which has been accepted for the award of any other degree or diploma in any university.

Signature:  Date: 1 Nov 2010

ABSTRACT

In recent years, a great deal of attention has been given to wireless connectivity solutions that are capable of establishing wireless ad-hoc networks between mobile nodes. Whilst most of these networks are formed using a combination of fixed and mobile infrastructure, completely infrastructure-less networks are thought to become more commonplace in the future. Moreover, this type of network structure seeks to utilise multi-hop connectivity between mobile nodes rather than the traditional single-hop connectivity established between fixed access points.

The initial configuration phase and subsequent maintenance phase of a multi-hop wireless ad-hoc network requires the use of appropriate routing functions to exist between the mobile nodes. Therefore, it is essential that a routing protocol capable of determining correct and optimal routing path information in the presence of node mobility and the mobile radio environment be sought. Furthermore, it is beneficial to utilise the limited wireless bandwidth efficiently, such that a routing protocol should be designed specifically in the context of a multi-hop wireless ad-hoc network topology. This can be achieved through employing a non-hierarchical approach and using neighbouring nodes to act as intermediate relay nodes.

The proposed routing protocol, called the Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP), is comprised of both a proactive and reactive routing component, thus forming a hybrid protocol which is able to exploit the benefits of each component. It is shown that manipulating these two components within the context of an awareness region, which divides the network into 2 regions, the routing overhead can be minimised. For the proactive component, the necessary network topology information that must be transmitted between neighbouring nodes is encoded within a routing update (RUPDT) packet. In this study, three alternative RUPDT encoding schemes have been formulated to encode the network topology in an efficient manner to reduce the RUPDT packet size.

For the reactive component, a novel covercasting mechanism is designed that minimises the number of route request (RREQ) transmissions required to determine the routing path by utilising existing routing table information. Supplementary techniques are then utilised, such as snooping, route repair, and route optimisation to further optimise performance and minimise the route discovery delay (latency). This same covercasting mechanism is then utilised to efficiently transmit periodic RUPDT packets between neighbouring nodes to maintain routing table validity at each node, without having to resort to flooding which causes the "broadcast storm problem". In addition, several route selection algorithms are considered which distribute traffic data between the intermediate relay nodes comprising the ad-hoc network.

The performance and computational complexity of the proposed hybrid routing protocol is shown by means of computer simulations and theoretical analysis. Various traffic scenarios and topologies are presented to obtain the routing protocol performance metric results, and these are compared with other protocols found in the literature. For a multi-hop wireless ad-hoc network, it is shown that the proposed hybrid routing protocol, MultiWARP, is able to achieve higher average system performance in terms of improved throughput and stability performance when compared to other wireless ad-hoc routing protocols, such as DSR.

ACKNOWLEDGMENTS

First and foremost, I hereby dedicate this PhD thesis to my father Ton Van Der Werf.

There are a number of people to whom I am indebted to for their assistance and support during this work that I would now like to acknowledge.

I am most grateful to the supervisors of this research, Professor Dr Kah-Seng Chung and Professor Dr Syed Islam. Their assistance, guidance, encouragement, and proof reading of my thesis have led me from the start to the completion of this work.

My gratitude to Dr Jing Xian Mei whose assistance and expertise with simulation techniques were invaluable to me.

Furthermore, I would also like to acknowledge my friends in the Communications Technology Research Group, namely, Dr Rajitha Palipana, Olusegun Aboaba, and Nick Jerrat, who provided a very friendly working environment.

Last but not least, I would like to thank my mother and father for their unceasing support, patience and loving encouragement which kept me focussed during this work.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

1G	First Generation
2G	Second Generation
2.5G	Second Generation Evolution
3G	Third Generation
3GPP	Third Generation Partnership Project
ACK	Acknowledgement
AMPS	Advanced Mobile Telephone Systems
AODV	Ad-hoc On-Demand Distance-Vector
AP	Access Point
ARM	Adapting to Route-demand and Mobility
ARP	Address Resolution Protocol
ARPAnet	Advanced Research Projects Agency Network
AS	Autonomous System
ASCII	American Standard Code for Information Interchange
BCP	Best Current Practice
BER	Bit Error Rate
BGP	Border Gateway Protocol
bps	Bits per second
BRP	Bordercast Resolution Protocol
BSR	Backup Source-Routing
BSSID	Basic Service Set Identification
C-ZRP	Caching Zone Routing Protocol
CADV	Congestion-Aware Distance-Vector
CBR	Constant Bit Rate
CCK	Complementary Code Keying
CDMA	Code Division Multiple Access
CLR	Clear Packet
CRC	Cyclic Redundancy Check
CSMA/CA	Carrier Sense Multiple Access with Collision Avoidance
CSMA/CD	Carrier Sense Multiple Access with Collision Detection

CTS	Clear To Send
CW	Contention Window
CWTS	China Wireless Telecommunication Standards
DATA	Data Packet
DB	Distributed Bordercast
DBPSK	Differential Binary Phase Shift Keying
DCF	Distributed Coordination Function
DDR	Distributed Dynamic Routing
DIFS	Distributed Inter-Frame Space
DQPSK	Differential Quadrature Phase Shift Keying
DSDV	Destination-Sequenced Distance-Vector
DSDV-SQ	Destination-Sequenced Distance-Vector Sequence Number
DSR	Dynamic Source Routing
DSSS	Direct-Sequence Spread Spectrum
DUAL	Diffusing Update Algorithm
EDGE	Enhanced Data Rates for Global Evolution
EGP	Exterior Gateway Protocol
EIFS	Extended Inter-Frame Space
EIGRP	Enhanced Interior Gateway Routing Protocol
ETSI	European Telecommunications Standards Institute
FC	Frame Control
FCS	Frame Check Sequence
FDD	Frequency Division Duplexing
FHSS	Frequency-Hopping Spread Spectrum
FIFO	First-In-First-Out
FSR	Fisheye State Routing
FTP	File Transfer Protocol
FZRP	Fisheye Zone Routing Protocol
GHz	Gigahertz
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GZRP	Geographical Zone Routing Protocol
HR-DSSS	High Rate Direct-Sequence Spread Spectrum
HSCSD	High Speed Circuit Switched Data

IANA	Internet Assigned Numbers Authority
IARP	Intra-zone Routing Protocol
IEEE	Institute of Electrical and Electronics Engineers
IERP	Inter-zone Routing Protocol
IGP	Interior Gateway Protocol
IGRP	Interior Gateway Routing Protocol
IMT	International Mobile Telecommunications
IP	Internet Protocol
IPv4	Internet Protocol Version 4
IPv6	Internet Protocol Version 6
IS-136	Interim Standard 136
IS-95	Interim Standard 95
IS-IS	Intermediate System to Intermediate System
ISM	Industrial Scientific and Medical
ITU	International Telecommunications Union
IZR	Independent Zone Routing
kbps	Kilobits per second
kHz	Kilohertz
LAN	Local Area Network
LSA	Link State Advertisement
MAC	Medium Access Control Layer
Mbps	Megabits per second
MCS	Mobile Control Station
MHz	Megahertz
MPDU	MAC Protocol Data Unit
MPR	Multipoint Relay
MS	MPR Selector
MSR	Multiple Source-Routing
MultiWARP	Multi-hop Wireless Ad-hoc Routing Protocol
MZRP	Multicast Zone Routing Protocol
NAV	Network Allocation Vector
NET	Network Layer
NMT	Nordic Mobile Telephone
NP	Nondeterministic Polynomial time

NS-2	Network Simulator 2
NTT	Nippon Telegraph and Telephone
OFDM	Orthogonal Frequency Division Multiplexing
OLSR	Optimised Link State Routing
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PHY	Physical Layer
PLCP	Physical Layer Convergence Procedure
PMD	Physical Medium Dependent
PN	Pseudo-Noise
QD1	Query Detection Level One
QD2	Query Detection Level Two
QoS	Quality of Service
R-DSDV	Randomised Destination-Sequenced Distance-Vector
RA	Receiver Address
RERR	Route Error
RFC	Request For Comments
RINC	Receiver Initiated NAV Clearing
RIP	Routing Information Protocol
RP	Routing Path
RQPD	Random Query Processing Delay
RREP	Route Reply
RREQ	Route Request
RTS	Request To Send
RUPDT	Route Update
RWP	Random Waypoint
SAP	Service Access Point
SBC	Selective Bordercasting
SC	Sequence Control
SFD	Start of Frame Delimiter
SIFS	Short Inter-Frame Space
SMS	Short Message Service
SNR	Signal to Noise Ratio
SNREP	SuperNode Reply

SNREQ	SuperNode Request
SOFDMA	Scalable Orthogonal Frequency Division Multiple Access
STA	Station
STD	Internet Standard
Sync	Synchronisation
TA	Transmitter Address
TACS	Total Access Communication System
TC	Topology Control
TCP	Transmission Control Protocol
TD-SCDMA	Time Division Synchronous Code Division Multiple Access
TDMA	Time Division Multiple Access
TIA	Telecommunications Industry Association
TTL	Time to Live
TZRP	Two-Zone Routing Protocol
UDP	User Datagram Protocol
UMTS	Universal Mobile Telecommunications Service
W-CDMA	Wideband Code Division Multiple Access
WAN	Wide Area Network
Wi-Fi	Wireless Fidelity
Wi-MAX	Worldwide Interoperability for Microwave Access
WLAN	Wireless Local Area Network
ZHLS	Zone-based Hierarchical Link State
ZRP	Zone Routing Protocol

LIST OF SYMBOLS

aCW_{max}	Contention window maximum value
aCW_{min}	Contention window minimum value
B_t	Random back-off time
C	Arbitrary timeout constant
D	Destination node
d	Distance in meters
δ	Value given by the previous segment's Neighbours field
$\delta()$	Edge weight of Bellman-Ford algorithm
d_{min}	Minimum hop distance
d_{prop}	Propagation delay between nodes
E, e	Edge
G	Graph
G_r	Receiver antenna gain coefficient
G_t	Transmitter antenna gain coefficient
h_r	Height of receiver antenna above ground
h_t	Height of transmitter antenna above ground
I	Identity matrix
i	Iteration step
I_i	Intermediate relay node
k	Edge weight of Dijkstra's algorithm
L	System loss factor
λ	Wavelength
M	Reachability matrix
μ	Periodic update interval
N	Number of vertices / Number of nodes / Number of hops
$n \times n$	Square matrix of size n
$O()$	Order of average computational complexity
p	Pause time
P_r	Received signal power
P_t	Transmitted signal power

$\boldsymbol{R}$	Radius of awareness region in hops
$\boldsymbol{R_i}$	Row index
$\boldsymbol{S}$	Source node
$\boldsymbol{T_i}$	Temporary matrices
$\boldsymbol{t_{limit}}$	Maximum period a node may remain in motion
$\boldsymbol{(u,v)}$	Current source vertex, Destination vertex
$\boldsymbol{v}$	Velocity to random waypoint
$\boldsymbol{V}$, $\boldsymbol{v}$	Vertex
$\boldsymbol{W}$	Temporary matrix

CHAPTER 1

INTRODUCTION

1.1 SCOPE OF THE PROJECT

In recent years, wireless ad-hoc networks have become commonplace throughout the world, mainly due to the proliferation of inexpensive wireless technology, such as IEEE 802.11 compliant devices (Hao et al. 2009; Hiertz et al. 2010). The IEEE 802.11 group of standards were originally designed for single-hop wireless connectivity, but due to popular demand, it had become the de-facto standard for multi-hop wireless ad-hoc networks (Xu & Saadawi 2001). Currently, the IEEE 802.11n specification is promoted as the latest revision of this technology, and is backwards compatible with the older IEEE 802.11b and 802.11g revisions. Furthermore, a draft specification for ad-hoc networks has been proposed called IEEE 802.11s but remains in development (IEEE Draft Std 802.11s 2010).

In a single-hop wireless ad-hoc network, the access point (AP) is responsible for the forwarding of packets to and from the mobile terminals, referred to as stations (STA). The AP is generally connected to an infrastructure, e.g., a fixed wired network, which can operate a number of different routing protocols, and will be described later in Chapter 3. The STA itself does not perform any routing, and hence does not require a routing protocol, as all packets are simply forwarded to the AP as the default gateway.

In a multi-hop wireless ad-hoc network, many autonomous STAs (also called nodes) form the foundation of a distributed network where fixed infrastructure is not present. As such, the routing functions provided by the AP for the single-hop case are not available, and therefore they must be provided by the nodes themselves. In this type of network the nodes are non-hierarchical in nature and primarily act as intermediate relay nodes, therefore, there is no default gateway for packets to travel towards. As a

result, a routing protocol for the node must be designed such that it supports the ad-hoc and multi-hop nature of the wireless network in which it operates. This can be achieved by communicating with the neighbouring nodes within the network, and by utilising intermediate relay nodes as gateways.

Recently, there has been an emergence of many ad-hoc routing protocols which aim at providing multi-hop connectivity using IEEE 802.11 infrastructure. There exist primarily two types of ad-hoc routing protocols, proactive and reactive protocols. Both types have their distinct advantages and disadvantages, and are discussed in greater detail in Chapter 4. The culmination of both proactive and reactive protocols results in the formation of a hybrid protocol, whose primary aim is to exploit the benefits of both types and combine them into a single protocol.

In this research, a hybrid routing protocol is proposed that incorporates the advantages whilst limiting the disadvantages of both the proactive and reactive routing protocols. This is achieved in two ways; firstly, by creating an arbitrarily defined awareness region which divides the network into 2 regions, and secondly, by utilising a route selection algorithm which minimises the number of route requests. The region within the awareness region is best suited with a proactive protocol as routing overhead efficiency decreases over distance. Thus by limiting the distance and size of the periodic routing update packets to be broadcast, efficiency can be maximised. For the region outside the awareness region a reactive protocol is best suited as routing overhead is minimal over longer distances. However, its inherent disadvantage is the delay incurred before a valid routing path is found.

In the proposed protocol, a "covercasting" mechanism is designed that minimises the number of route request transmissions needed to determine the routing path by utilising existing routing table information. Furthermore, the route request packets are guided towards key nodes which most likely have a valid routing path to the destination. This in effect minimises the route discovery delay (latency), whilst simultaneously reducing the routing overhead compared to other ad-hoc routing protocols.

1.2 OBJECTIVES & ORIGINAL CONTRIBUTIONS

The primary objectives of this research are:

- To design a hybrid routing protocol suitable for a multi-hop wireless ad-hoc network that provides a suitable routing path in an efficient and timely manner.
- To design a proactive routing component that transmits routing update packets periodically without flooding of the network, whilst minimising overhead in terms of awareness region size and packet size.
- To design a reactive routing component that exploits the routing information provided by the proactive component in order to minimise and guide route request packets.
- To formulate a fail-safe routing scheme whereby routing paths can undergo route repair after an unexpected link breakage has occurred, or alternatively, optimised on-the-fly if shorter alternative routing paths can be found.
- To evaluate the performance and computational complexity of the proposed hybrid routing protocol using computer simulations and theoretical analysis.

In achieving these aims, the following novel contributions are highlights of this research:

A new hybrid routing protocol, called the Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP), has been designed and is proposed for use in multi-hop wireless ad-hoc networks. The MultiWARP protocol incorporates the strengths of both proactive and reactive routing protocols, and does not require the use of the address resolution protocol (ARP), which significantly decreases routing path acquisition time. It is shown that MultiWARP discovers and acquires a valid routing path, as well as alternative backup routes, for an outgoing packet with minimal delay. This is discussed in Sections 6.6 and 7.3.4.

In an effort to prevent the “broadcast storm problem”, and its associated flooding and channel contention problems, a mechanism which limits the distance a routing

update packet is propagated is employed. To minimise routing overhead, the topological connectivity information encapsulated within the routing update packet is limited to an awareness region of R hops for the proactive routing component. Through careful encoding of vital network topology information, such as node connectivity and routing metrics, the size of the route update packet is minimised using three alternative encoding schemes. Furthermore, other avenues of accumulating routing information through snooping packets that are traversing the network, and the use of a novel SuperNode mechanism to obtain a larger awareness region is provided. These contributions are discussed in detail in Chapter 5.

For the reactive routing component, a route request algorithm is proposed that constructs and solves the reachability matrix which contains the network topology information. The algorithm calculates the smallest subset of nodes, from a set of possible candidate nodes, to which the route request packet is guided to. MultiWARP then targets these specific nodes (termed covercasting) to facilitate the reduction of routing overhead by minimising the number of nodes that the route request packet must be transmitted to. This minimisation is performed through exploiting information contained within the local routing table, and through application of the NP-complete Set Covering Problem. The proposed algorithm has an average computational complexity characterised as $O(\log n)$. These further contributions are presented in Chapter 6.

1.3 STRUCTURE OF THE THESIS

The background information for this thesis is presented in Chapter 2. This includes a brief history of wireless communications, and a discussion on the physical (PHY) and medium access control (MAC) layers that control the utilisation of the wireless channel. The data transmission sequence as well as a novel snooping mechanism of the handshaking procedure is then presented.

A thorough review of routing protocol fundamentals, organisation, and classes are provided in Chapter 3. Furthermore, an extensive review of proactive, reactive, and hybrid ad-hoc routing protocols are provided in Chapter 4. Particular emphasis is

given on flat routing protocols such as the destination-sequenced distance vector (DSDV), optimised link state routing (OLSR), dynamic source routing (DSR), and the hierarchical zone routing protocol (ZRP).

The design of the proposed hybrid routing protocol, called Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP), is provided in Chapter 5 and Chapter 6. The proactive component is the focus of Chapter 5, and discusses the construction of the periodic route update packet, including its associated update interval and routing path snooping capability. Three alternative route update packet encoding schemes have been formulated and compared in terms of overall packet size. These encoded packets are then decoded to reproduce the network topology. Furthermore, a novel SuperNode mechanism is described whereby selected nodes can carry extended routing table information by utilising different sized awareness regions.

In Chapter 6, the reactive component is presented which aims at minimising the number of route request packets transmitted using a proposed covercasting mechanism. A detailed description of the construction of the reachability matrix from given network topology is provided, including the construction of the route request, route reply, and route error packets. Furthermore, an algorithm is presented to solve the reachability matrix, such that the route request packet is transmitted to the minimal set of nodes required for route discovery. An analytical model for the computational complexity of the proposed routing protocol is provided. In addition, the structure of the routing table is presented, as well as route selection, repair, and optimisation strategies to optimise performance.

The results from the computer simulations and performance evaluations are assessed and presented in Chapter 7. The impact of various traffic scenarios and topologies are examined, and the results are presented in terms of overall performance metrics, including: packet delivery ratio, traffic throughput, delay characteristics, packet losses, route discovery latency, routing overhead, route selection, and routing stability. In conclusion, the major findings are reviewed and recommendations for future studies are presented in Chapter 8.

CHAPTER 2

WIRELESS COMMUNICATIONS

2.1 INTRODUCTION

The development of wireless communication was started by Nikola Tesla in 1893 through his invention of a device capable of transmitting and receiving electromagnetic energy (Cheney 2001). This in effect was the creation of the world's first radio transceiver. Two years later in 1895, Gugleilmo Marconi demonstrated the ability to transmit and receive telegraphic messages over a distance wirelessly (Cheney 2001). This can be said to be the world's first "digital" communication as the message was coded in Morse code, which was originally developed by Samuel Morse in the 1840s for the transmission of messages over metallic cables (ITU 2004).

Since then, much advancement has been made in the field of communications, including the first wireless voice communication in 1900 by Reginald Fessenden (Belrose 2002), to the creation of the packet-switched ARPAnet in 1969 which was the predecessor to the Internet (Salus 1995). The introduction of cellular wireless communications in 1979 has involved a migration from the circuit switched solutions of the first and second generation networks in 1990s, towards packet switching in third generation networks in early 2000s.

2.2 WIRELESS LAN (WI-FI)

Through the culmination of wired technology, such as IEEE 802.3 Ethernet (IEEE Std 802.3 2005), and wireless CDMA technology, a wireless networking system has emerged called a wireless local area network (WLAN) (Agrawal et al. 2008). The carrier sense multiple access with collision avoidance (CSMA/CA) protocol in

WLAN is very similar to the carrier sense multiple access with collision detection (CSMA/CD) protocol in IEEE 802.3 Ethernet (Fourty et al. 2005). A WLAN, which is also referred to as Wi-Fi, has been developed by the Wi-Fi Alliance and is based primarily on the IEEE 802.11(a,b,g,n) standards. This technology utilises the 2.4 and 5.0 GHz Industrial Scientific and Medical (ISM) unlicensed bands.

The distinction with Wi-Fi is that it has a high throughput speed but with a reduction in transmission range when compared to Wi-MAX, which is aimed primarily at high speed long distance transmission. For example, an IEEE 802.11g based Wi-Fi is capable of a maximum theoretical data rate of 54 Mbps over a maximum 250 m range, in comparison with a Wi-MAX maximum theoretical data rate of 70 Mbps over a maximum 50 km range (Fourty et al. 2005). It should be noted that these values are trade-offs, such that the maximum throughput is not achievable at the maximum range due to signal deterioration and interference, and these high throughput rates are rarely achieved in practice. Moreover, the Wi-Fi MAC revolves around a channel contention process, discussed in detail in Section 2.5, whereas the Wi-MAX MAC uses a more complex scheduling algorithm that supports guaranteed quality-of-service (QoS) (IEEE Std 802.16 2004). A comprehensive report of the similarities and differences are given by Fourty (Fourty et al. 2005), and is summarised in Table II therein.

A WLAN is primarily used to extend an existing wired network, namely an IEEE 802.3 Ethernet LAN, through the introduction of an IEEE 802.11 based wireless access point (AP) (IEEE Std 802.11 1999). In this thesis, the IEEE 801.11b physical (PHY) layer is used as the air interface for the communication environment under consideration. The more recent IEEE 802.11n PHY layer (IEEE Std 802.11n 2009) differs from the 802.11b PHY layer, but this does not impact on the design or operation of the hybrid routing protocol proposed in this thesis as it resides within the network (NET) layer. The physical layer and medium access control layer of IEEE 802.11b will be discussed in greater detail in Sections 2.4 and 2.5, respectively.

2.3 OSI MODEL AND TCP/IP STACK

The design of a hybrid routing protocol must address the benefits and shortcomings of cross-layer protocol design. Accordingly, it is suitable to introduce two distinct communication system models which illustrate the layered approach and describe the functions of each respective layer. The open systems interconnection (OSI) model (Zimmermann 1980) is an abstract 7-layer structure of a communication system from the physical hardware to the software application, as shown in Table 2.1.

Layer	Name	Protocols	Description
1	Physical Layer	802.11 PHY	Modulation, data frame structures, carrier sensing.
2	Data Link Layer	802.11 MAC	Node to node communication.
3	Network Layer	IP, ARP, Routing	Path discovery, logical addressing.
4	Transport Layer	TCP, UDP	Connection(less) traffic flow.
5	Session Layer	Sockets	Connection status management.
6	Presentation Layer	ASCII	Data representation and conversion.
7	Application Layer	FTP, Telnet	Software application functions.

Table 2.1 – The 7-layer open systems interconnection (OSI) model of a communication system. The protocols listed are used only as examples for each layer.

The interface between layers allows for a structured design of a given system, and thus there exists a two-way flow of information between adjacent layers. The lower layers provide a service for the upper layers, however, it should be noted that not all layers are necessary for a given communication system. Furthermore, precise separation between layers is sometimes not clearly evident. This is particularly the case for cross-layer designs involving multiple service types per layer.

Another conventional model for a communication system is the 4-layer transmission control protocol / Internet protocol (TCP/IP) stack (Braden 1989), as shown in Table 2.2. The TCP/IP stack is an informal, less abstract model than the OSI model such that each layer does not have a well-defined protocol specification. As such, the

layering of a communication system is deemphasised to highlight the importance of cross-layer design. In some applications, another layer may be added to the TCP/IP stack, namely the physical layer, in order to make it analogous with the OSI model.

Layer	Name	Protocols	Description
- / (1)	Physical Layer	802.11 PHY	Signal modulation, carrier sense
1 / (2)	Link Layer	802.11 MAC	Node to node communication
2 / (3)	Network Layer	IP, ARP, Routing	Path discovery, logical addressing
3 / (4)	Transport Layer	TCP, UDP	Connection(less) traffic flow
4 / (5)	Application Layer	FTP, Telnet	Combination of OSI Layers 5 – 7.

Table 2.2 – The 4-layers of the TCP/IP stack. An extra layer analogous to the OSI Layer 1 is added to make it a 5-layer stack, indicated by brackets.

2.4 PHYSICAL LAYER

A number of modulation schemes are included in the IEEE 802.11 specifications for controlling the transmission and reception of data frames, and belong to Layer 1 of the OSI model. The specified modulation types are direct-sequence spread spectrum (DSSS), frequency-hopping spread spectrum (FHSS), orthogonal frequency division multiplexing (OFDM), and a combination of these schemes. In this thesis, the DSSS modulation type as used in the IEEE 802.11b standard is considered.

The spread spectrum modulation scheme employed by the PHY layer is governed by a physical medium dependent (PMD) procedure. The principle of the spread spectrum modulation technique is through the use of spreading sequences such that the transmitted bandwidth of a signal is considerably increased. Using this technique, the energy of a signal is dispersed or “spread out” over a large section of the allocated frequency spectrum. The type of spreading sequences used in DSSS are also referred to as channelisation codes, which are comprised of a sequence of pseudo-noise (PN) random bits (Proakis & Salehi 2001). This code is used to spread the data bits within the data frames provided to the PHY layer. In IEEE 802.11b, the

data in a frame is spread using an 11-chip Barker sequence as given below (IEEE Std 802.11 1999):

+1, –1, +1, +1, –1, +1, +1, +1, –1, –1, –1.

2.4.1 Data Framing

The framing of data is governed by the physical layer convergence procedure (PLCP) which is supplied with data from the medium access control (MAC) layer in the form of a MAC protocol data unit (MPDU). This yields a PHY layer frame of data independent of the MAC layer data, and therefore the modulation scheme can then be employed to convert the frame into a signal suitable for transmission on the wireless medium. The PLCP frame format is shown in Figure 2.1.

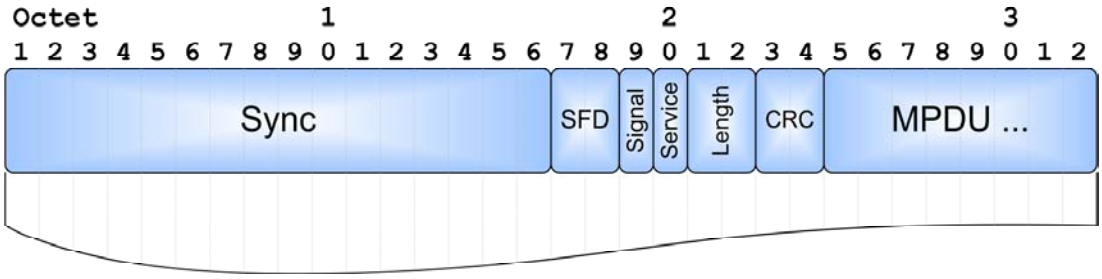


Figure 2.1 – PLCP frame format consisting of the PLCP preamble (*Sync*, *SFD*), and the PLCP header (*Signal*, *Service*, *Length*, *CRC*), and the *MPDU* field which is used to store and transmit the data supplied by the MAC layer.

The reception of frames uses the reverse process of transmitting, such that MPDUs are created by the PLCP from incoming frames of data and passed to the MAC layer. The communication between the PHY and MAC layer is governed by an abstract entity called the service access point (SAP) which provides communication primitives, and is termed the PHY-SAP interface (IEEE Std 802.11 1999). The reader is referred to Chapter 15 “DSSS PHY Specification” of the IEEE reference for further details.

2.4.2 Modulation of Data Frame

The data rate for modulation used in IEEE 802.11b is referred to as the basic data rate (1 Mbps) and enhanced data rate (2 Mbps), and uses differential binary phase shift keying (DBPSK) and differential quadrature phase shift keying (DQPSK), respectively. A high-rate DSSS (HR-DSSS) specification also allows for a 5.5 Mbps and 11 Mbps data rate, and requires the use of an 8-chip complementary code keying (CCK) modulation scheme. The allocated frequency spectrum for operation is 2.4–2.4835 GHz, which is divided into 14 channels, as shown in Figure 2.2.

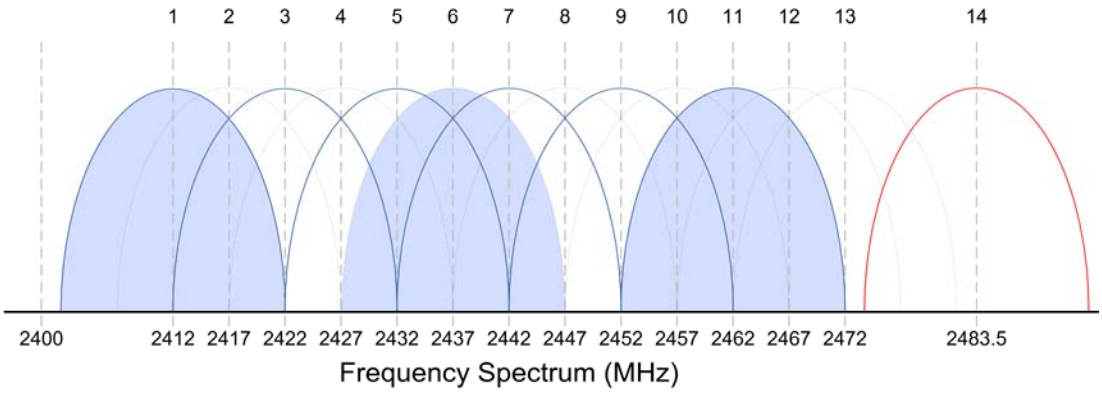


Figure 2.2 – Frequency spectrum of the IEEE 802.11b standard showing the channels spaced at 5 MHz intervals (channels 1-13) and 11.5MHz for channel 14 distributed within the allocated 2.4–2.4835 GHz band.

As the channel bandwidth is 22 MHz, these channels can be subdivided into 3 non-overlapping or 6 overlapping channels. The blue shaded areas represent the 3 non-overlapping channel selections given the 22 MHz channel bandwidth requirement. The 6 overlapping channel selections are represented by the solid blue arc and are available in Australia, North America, and most of Europe (excluding Spain and France). The red arc represents the additional channel 14 which is available in some countries such as Japan. In this thesis, the enhanced data rate (2 Mbps) is used for data transmission, and the basic data rate is used for the contention process (1 Mbps). This is done in order to keep performance and results comparisons as close as possible to other research in the field.

2.4.3 Simulated Radio Propagation Model

In this thesis, the technique used to replicate the behaviour of the above mentioned IEEE 802.11b physical layer specification is through software simulation. The NS-2 simulation environment software (NS-2 2005) is used to simulate the proposed routing protocol, and this requires a radio propagation model to mimic the physical layer. This characterises the physical layer in terms of received signal power for each packet received at a particular node. The received signal power is calculated by the radio propagation model and determines if the power is sufficient for packet reception (i.e., the power is above receive threshold), or if the packet should be dropped. There exist three radio propagation models within NS-2, namely, a free-space model, a two-ray ground reflection model, and a shadowing model.

The free-space model assumes only line-of-sight communication exists between the transmitter and receiver pair, and is characterised by equation 2.1 for the received power, P_r, at the receiver being d meters away from the transmitter (NS-2 2005)

$$\mathrm{P_r}(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \qquad \text{(Eq. 2.1)}$$

where P_t is the transmitted signal power, G_t and G_r are the coefficients of antenna gain, λ is the wavelength, and L is the system loss factor. In NS-2, the coefficients of antenna gain and system loss are equal to 1, thus yielding equation 2.2.

$$\mathrm{P_r}(d) = \frac{P_t \lambda^2}{(4\pi)^2 d^2} \qquad \text{(Eq. 2.2)}$$

An extension to the free-space model is the two-ray ground reflection model, which includes the line-of-sight and a ground reflected path. The received power, P_r, with the receiver being d meters away from the transmitter is given by (NS-2 2005)

$$\mathrm{P_r}(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L} \qquad \text{(Eq. 2.3)}$$

where h_t and h_r are the heights of the antennas above ground for the transmitter and receiver, respectively. As this model suffers from oscillation at short distances (NS-2 2005), a cut-off point is established whereby the free-space model is used before, and thereafter the two-ray ground reflection model is used. The cut-off point (in terms of meters away from the transmitter) is given by equation 2.4.

$$d_c = \frac{4\pi h_t h_r}{\lambda} \qquad \text{(Eq. 2.4)}$$

The shadowing model is used when the receiver is located in a shadow from the transmitter. This model uses multi-path fading as another component in determining the received power and thus yields a more realistic radio propagation model (NS-2 2005). In this thesis, a more complex radio propagation model for simulation is not necessary and would cause for inconsistent comparisons with other research in the field, thus the two-ray ground reflection model is chosen.

2.5 MEDIUM ACCESS CONTROL LAYER

The medium access control (MAC) layer is responsible for the coordinated sharing and utilisation of the wireless channel, and is defined on Layer 2 of the OSI model and Layer 1 of the TCP/IP stack. In this thesis, the IEEE 802.11b standard is used and the coordinated sharing is through use of a distributed coordination function (DCF) which relies on a carrier sense multiple access with collision avoidance (CSMA/CA) scheme. The DCF utilises a physical carrier sense (signal sensing) provided by the PHY layer and a virtual carrier sense provided by the MAC layer, which is termed the network allocation vector (NAV). The NAV is a variable which indicates whether or not a node is allowed to contend for channel access. This is based on the reception of signalling information (e.g., the *Duration* field) within the request-to-send (RTS) and clear-to-send (CTS) packets, as discussed in Section 2.5.1. When the NAV indicates the channel is idle, a node may contend for access to the channel, otherwise, it must initiate the deferring mechanism, which is discussed in Section 2.5.3.

2.5.1 Data Transmission Sequence

The data transmission sequence consists of an RTS packet followed by a CTS reply packet, which in turn is followed by a data packet and finally concluded by an acknowledgement (ACK) packet, termed the RTS-CTS-DATA-ACK sequence. The scheduling of the data transmission sequence is according to Figure 2.3, and the four packet types mentioned, including the timing intervals, are discussed below.

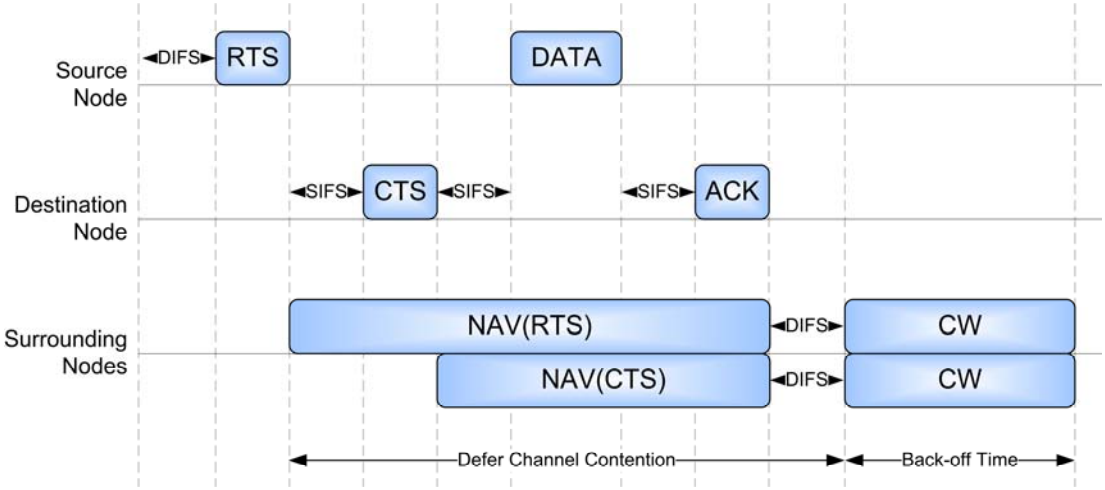


Figure 2.3 – The network allocation vector (NAV) settings for various stages of the data transmission sequence as defined by the RTS-CTS-DATA-ACK sequence. This figure is adapted from the IEEE 802.11b standard (IEEE Std 802.11 1999).

The IEEE 802.11 parameter *dot11RTSThreshold* dictates when the RTS-CTS sequence is activated, and can be set using the "*Mac/802_11 set RTSThreshold_*" parameter in the simulation environment. According to IEEE 802.11b, it can be set between zero and 2312 bytes, and in this thesis the value is set to zero bytes.

It should be noted that for data packets smaller than *dot11RTSThreshold*, an RTS-CTS scheme creates more overhead in terms of channel utilisation than the time it takes to transmit the data packet. Therefore, the basic access method is preferred which does not require any RTS, CTS or ACK packet. In this case, the channel is sensed idle for a minimum period of time and the data packet is transmitted on the channel immediately thereafter. If the channel is sensed to be busy, the network

allocation vector (NAV) deferring mechanism is activated. The NAV is set when a node has to defer channel contention, after which, a distributed inter-frame space (DIFS) interval and a random back-off time period must elapse before the node is allowed to contend for channel access. The minimum period of time and the random back-off scheme is discussed later in Section 2.5.3.

In this thesis, the basic access method is not utilised, instead, the RTS-CTS-DATA-ACK ("handshaking") method is adopted in order to keep the integrity between comparisons of results in other published literature using the handshaking method.

2.5.1.1 Request to Send (RTS)

The RTS is an indication by a source node to signify it wants to transmit data on the channel, where the RTS packet structure is shown in Figure 2.4. The RTS packet contains information relating to the duration of transmission, the source and destination node MAC addresses, as well as a frame check sequence (FCS) which contains a cyclic redundancy check (CRC) checksum and a frame control (FC) code.

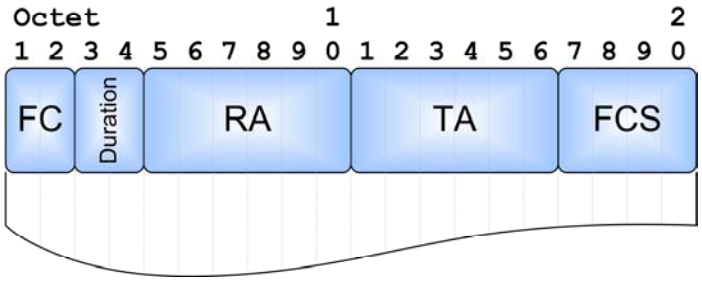


Figure 2.4 – RTS Packet Structure showing the *frame control* (*FC*) code, the *Duration* field which indicates the time it will take for the following data transmission sequence, the MAC *receiver address* (*RA*), the MAC *transmitter address* (*TA*), and the *frame check sequence* (*FCS*) cyclic redundancy check value.

The *FC* code contains various protocol version identification numbers, frame management codes, and power control information and always precedes all RTS-CTS-DATA-ACK packets. The *Duration* field of the RTS packet specifies the time in microseconds required to transmit the intended data packet, a CTS packet, an

ACK packet and three short inter-frame space (SIFS) intervals, as discussed later in Section 2.5.3. The source and destination node MAC addresses are termed *transmitter address* (*TA*) and *receiver address* (*RA*), respectively. The *FCS* field is always appended to all RTS-CTS-DATA-ACK packets in order to confirm the integrity of the received packets in terms of checking the CRC value.

The RTS packet is transmitted by the source node in a two-step process. The channel must be sensed idle for a period of time termed the contention period, and is defined by the distributed inter-frame space (DIFS) interval, as discussed later in Section 2.5.3. If the channel is sensed as being idle during the contention period, the second step is to immediately transmit the RTS packet on the channel. Following the RTS packet transmission, the surrounding neighbouring nodes will set their NAV to reserve the channel, termed NAV(RTS), which allows time for the data packet to be transmitted by the source node. If the channel was sensed to become non-idle (or busy), then the RTS packet is not transmitted. Instead, the source node will defer its contention of the channel until once again the channel becomes idle, and repeat the contention process described above.

Since collision detection is not possible during transmission over a wireless channel using the CSMA/CA technique, the source node will only know that the transmission of an RTS packet was successful based on the reception of a CTS reply packet from the destination node. In the case of a collision at the destination node between the transmitted RTS_1 and another packet, for example interference with another RTS_2, it is likely that the RTS_1 packet will be corrupted at the destination node.

When the CTS reply is not received after some time termed the *CTSTimeout* interval, given by equation 2.5, the source node will assume RTS reception failure. The variables *SIFS* and *aSlotTime* are defined by the PHY layer characteristics given in Table 2.3 in Section 2.5.3.

$$CTSTimeout = SIFS + ACK_{tx\_duration} + aSlotTime \qquad \text{(Eq. 2.5)}$$

The RTS packet can then be retransmitted by the source node until the number of retransmission attempts equals *dot11ShortRetryLimit* (default of 7), after which the corresponding data packet is dropped. At this stage the MAC layer no longer assumes responsibility for the data packet, and it is up to the upper layers (e.g., the routing or transport layer) to rectify the fault.

2.5.1.2 Clear to Send (CTS)

The CTS packet is transmitted as a response to the successful reception of an RTS packet, and is an indication of a willingness to accept a data packet, where the CTS packet structure is shown in Figure 2.5. The primary role of the CTS packet is to inform the source node that the destination node is ready, and that it can start to transmit the data packet. The secondary role of the CTS packet is to inform the neighbouring nodes around the destination node to remain silent for the duration of the ensuing data transmission by setting its NAV to busy, which in effect acts as a type of a one-time channel reservation technique.

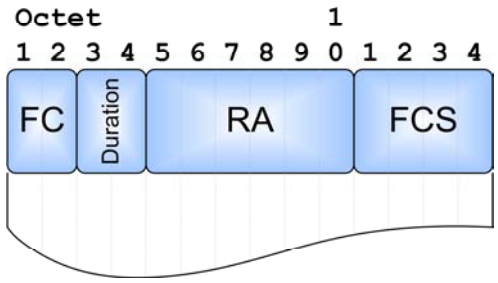


Figure 2.5 – CTS Packet Structure showing the *frame control* (*FC*) code, the *Duration* field which indicates the time it will take for the following data transmission sequence, the MAC *receiver address* (*RA*), and the *frame check sequence* (*FCS*) cyclic redundancy check value.

The *FC* code and *FCS* are the same as for the RTS packet structure discussed above, and the *Duration* field is copied from the RTS packet and adjusted by subtracting the time in microseconds required to transmit the CTS packet and a SIFS interval. The *RA* field is copied from the *TA* field from the RTS packet.

The CTS packet is transmitted in a similar two-step manner to the RTS packet transmission process. Firstly, the channel must be sensed idle for a period of time, but only for a SIFS interval as opposed to a longer DIFS interval for the RTS packet. If the channel is sensed as being idle during the entire period, the second step is to immediately transmit the CTS packet on the channel. Following the CTS packet transmission, the surrounding neighbouring nodes will set their NAV to reserve the channel, termed NAV(CTS), which allows time for the data packet to be transmitted by the source node. If the channel was sensed to become non-idle (or busy), then the CTS packet is not transmitted. In contrary to the RTS process, the destination node will not defer its contention of the channel, and will not repeat the contention process in order to retransmit a CTS packet. Therefore, there is no CTS retransmission if it was unsuccessful in being received by the source node.

If the data packet is not received after some time by the destination node, termed the *Resetting the NAV* period, the destination node will not inform the source node of the failure, and the destination node will be allowed to contend for channel access, and the NAV is cleared. The *Resetting the NAV* period is made possible by PHY carrier sensing, such that the NAV is able to be cleared if no transmission has occurred on the channel during a defined period. This period is the time in microseconds required to transmit the CTS packet, two SIFS intervals, and two aSlotTimes, as discussed later in Section 2.5.3. The *Resetting the NAV* period is commenced from the time the RTS packet has been received.

The neighbouring nodes that also received the CTS packet will not be able to detect that the data packet is not being received by its intended destination node, and is therefore unable to clear the NAV. This results in these neighbouring nodes having to unnecessarily defer channel contention, and thus causing under-utilisation of the wireless channel. One type of clearing mechanism is proposed by Du, et al. (Du & Chen 2005) whereby the destination node transmits a clear (CLR) packet to its surrounding neighbours (the same neighbours that received the CTS packet) after its *Resetting the NAV* period has expired. This results in those neighbouring nodes to also be able to reset their NAV because they are unable to use PHY carrier sensing, and results in better utilisation of the wireless channel. The concept of clearing is discussed in further detail in Section 2.5.2.

2.5.1.3 Data Packet (DATA)

The data packet is transmitted as a response to the successful reception of a CTS packet, and is an indication by the destination node to start transmission of the data packet, where the data packet structure is shown in Figure 2.6. The primary role of the data packet is to encapsulate the data from higher layers (e.g., the transport or application layer) to provide a flow of information between the source and destination node.

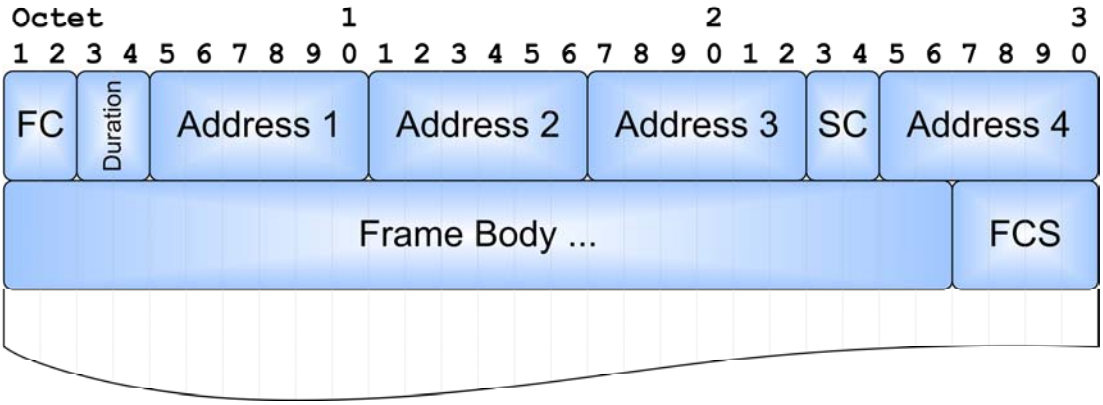


Figure 2.6 – Data Packet Structure showing the *frame control* (*FC*) code, the *Duration* field which indicates the time it will take for the following data transmission sequence, the four *Address* fields and the *Sequence Control* (*SC*) field are dependent on the basic service set identification (BSSID) being used, the *Frame Body* contains the encapsulated data, and the *frame check sequence* (*FCS*) cyclic redundancy check value.

The *FC* code and *FCS* are the same as for the RTS and CTS packet structures discussed above, and the *Duration* field specifies the time in microseconds required to transmit an ACK packet and a short inter-frame space (SIFS) interval for unfragmented data transmission. The *Address* and *sequence control* (*SC*) fields are dependent on the basic service set identification (BSSID) being used, and is beyond the scope of this thesis. For the general case, *Address 1* is equivalent to the *RA* and *Address 2* is equivalent to the *TA*, where the *Frame Body* field contains the actual encapsulated data. For further information on BSSID codes, the reader is referred to Section 7.2.2 of the IEEE 802.11b specification (IEEE Std 802.11 1999).

The data packet is transmitted in a one-step manner, which is different to the two-step RTS and CTS packet transmission process. For data packet transmission, the channel does not have to be sensed idle for a period of time. Instead, the source node only waits for a SIFS interval to elapse and proceeds to transmit the data packet immediately thereafter. If the channel was sensed to become non-idle (or busy), then the data packet is still transmitted regardless of this fact.

2.5.1.4 Acknowledgement (ACK)

The ACK packet is transmitted as a response to the successful reception of a data packet, and signifies a positive acknowledgement by the destination node, where the ACK packet structure is shown in Figure 2.7.

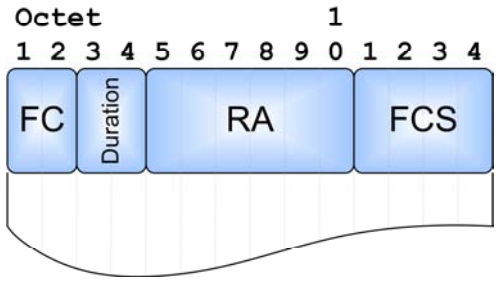


Figure 2.7 – ACK Packet Structure showing the *frame control* (*FC*) code, the *Duration* field which is usually set to zero (unless fragmented data is being used), the MAC *receiver address* (*RA*), and the *frame check sequence* (*FCS*) cyclic redundancy check value.

The *FC* code and *FCS* are the same as for the RTS, CTS and data packet structures discussed above, and the *Duration* field is set to zero as it is the final packet in the RTS-CTS-DATA-ACK sequence.

The ACK packet is transmitted in a one-step manner identical to the data packet transmission process, as discussed in Section 2.5.1.3. The source node that is awaiting the arrival of the ACK packet continues to wait for an *ACKTimeout* interval, given by equation 2.6, before indicating a data transmission sequence failure

to the MAC layer. The variables *SIFS* and *aSlotTime* are defined by the PHY layer characteristics given in Table 2.3 in Section 2.5.3.

$$ACKTimeout = SIFS + ACK_{tx\_duration} + aSlotTime \qquad \text{(Eq. 2.6)}$$

After this interval has elapsed, the source node may attempt retransmission of the data packet up to a maximum of *dot11LongRetryLimit* (default of 4) if the data packet is larger than *dot11RTSThreshold*, or a maximum of *dot11ShortRetryLimit* (default of 7) if the data packet is smaller than *dot11RTSThreshold* (IEEE Std 802.11 1999; Manshaei et al. 2005).

2.5.2 Snooping of RTS-CTS Packets

From the 6-node transmission range model shown in Figure 2.8, it can be visualised that when a certain node transmits information, not only does the intended recipient node receive the packet, but also other surrounding nodes. The overhearing of RTS and CTS packets not directly intended for a specific node is termed snooping.

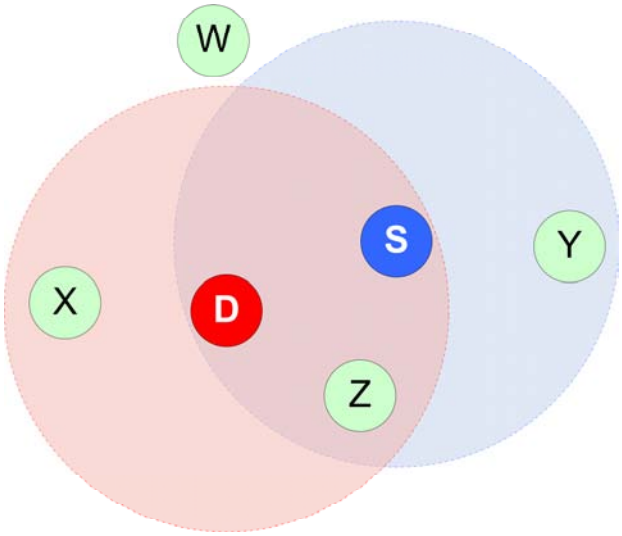


Figure 2.8 – The 6-node transmission range model showing 6 nodes at various distances from the source node **S** and destination node **D**. The 4 surrounding nodes **W**, **X**, **Y**, and **Z** comprise all possible combinations of being within range of either node **S** or node **D**. This represents the ability for a surrounding node to receive or snoop an RTS or CTS packet.

Given Figure 2.8, when the source node **S** transmits an RTS packet to the destination node **D**, both nodes **Y** and **Z** receive the RTS packet. In this situation, node **X** is known as the "hidden node" because if it transmits during the communication between nodes **S** and **D**, it will result in a collision. Similarly, when node **D** reacts by transmitting a CTS packet back to node **S**, both nodes **X** and **Z** receive the CTS packet. In this case, node **Y** is known as the "exposed node" because it is not able to interfere with the communication between nodes **S** and **D**. This is because the received signal strength at node **D** from node **S** $\gg$ node **Y**, and thus node **Y** remains silent for no reason during the subsequent DATA packet transmission.

The snooping of these packets by other nodes is a desirable and a necessary trait for the RTS-CTS handshake to work. Ultimately, the more important of the two packets is snooping of the CTS packet; however, snooping of the RTS packet is also required to make snooping of the CTS packet successful. This can be appreciated by knowing that the nodes surrounding the RTS transmitter must remain idle in order for node **S** to receive the CTS reply. However, there are some undesirable characteristics that emerge from snooping, and these are discussed hereafter.

The standard method by which the RTS-CTS handshake operates is through snooping, and thereby it is able to inform the MAC layer of the surrounding nodes not to contend for channel access for a certain amount of time specified in the RTS packet structure. The nodes around node **S** that have snooped the RTS information, in this case nodes **Y** and **Z**, will remain silent for the requested time period by setting their NAV(RTS) accordingly. Two scenarios can occur which demonstrate the undesirable characteristics of snooping.

In the first scenario, suppose that node **D** did not receive the RTS, but nodes **Y** and **Z** did. Since node **D** has no prior knowledge that an RTS is coming, it will therefore not transmit the CTS packet. However, both nodes **Y** and **Z** will set their NAV(RTS) timer accordingly and will remain idle for the requested amount of time to complete the transmission sequences. Obviously, the data packet will never be sent by node **S** due to no CTS response from node **D**. Therefore, it can be concluded that both nodes **Y** and **Z** are being unnecessarily silent for the entire requested period.

In the second scenario, suppose that node **D** did receive the RTS, and is proceeding to transmit the CTS reply. Now suppose that for some reason the CTS does not arrive successfully at node **S**, but does arrive at nodes **X** and **Z**. Accordingly, both nodes **X** and **Z** will set their NAV(CTS) timer and remain idle for the requested amount of time. Again, it is evident that the data packet will not be transmitted by node **S** due to not receiving a CTS response from node **D**. Therefore, from the two scenarios it can be concluded that all three nodes **X**, **Y**, and **Z** will remain unnecessarily silent for the entire requested period, whilst the source and destination nodes will not be transmitting in either event.

To counter the problem associated with the first scenario, we can make use of some additional information exhibited in the situation and by introducing a modified NAV. It is evident that the nodes who snooped the RTS packet, namely nodes **Y** and **Z**, will also be able to observe the data packet that will be transmitted by node **S**. In this case, nodes **Y** and **Z** should set their NAV to indicate which nodes were involved in the RTS-CTS handshake by setting their NAV(RTS[S,D]). Now, from nodes **Y** and **Z** point of view, if the data packet is not observed, it can be assumed that either the RTS was not successful at reaching node **D**, or the CTS was not successful at reaching node **S**. In either case, node **S** will not transmit the data packet.

In order for nodes **Y** and **Z** to detect whether or not node **S** is transmitting the data packet, nodes **Y** and **Z** must observe the channel for a small amount of time, similar to the *Resetting the NAV* period discussed in Section 2.5.1.2. This period should be long enough to receive the header fields of the data packet, and determine if the source and destination nodes match the RTS information received through snooping, i.e., NAV(RTS[S,D] matches the current transmission. At this point, if the start of the data packet was not received, or if the information obtained from the header of the data packet does not match the RTS information, the MAC layer will decide that nodes **Y** and **Z** no longer have to remain silent for the entire requested period. For this reason, nodes **Y** and **Z** will be allowed to clear their NAV and be allowed to contend for access to the channel.

To counter the second scenario, a slightly more complicated approach is required. In this case, nodes **X** and **Z** were able to observe the CTS reply packet from node **D**,

causing their NAV(CTS[D,S]) to be set. However, node **S** never received the CTS successfully and thus node **S** will not transmit the data packet. However, unlike before, we cannot assume that nodes **X** and **Z** are able to observe the data packet. From the topology in Figure 2.8, node **Z** will snoop both the RTS and CTS packets, but node **X** will only snoop the CTS packet. This being the case, node **X** will never be able to observe the data packet since it is out of range of node **S**, which accounts for why it was unable to snoop the RTS packet. The solution from the first scenario by observing the channel for a certain period of time will not work because even if the data packet is transmitted, node **X** will decide that nothing is being received causing it to timeout. This means, node **X** will incorrectly clear its NAV, and be able to contend for channel access, which could cause a collision at node **D**.

The solution calls for an additional packet to be transmitted by node **D**, which is in range of the data packet, and hence node **D** can determine whether or not the data packet is being transmitted by node **S**. If node **D** decides that the data packet was not received, or if the information obtained from the header of the data packet does not match the RTS information, then node **D** will transmit a clear (CLR) packet to its surrounding neighbours. The CLR packet is transmitted before the *Resetting the NAV* timer is allowed to expire, such that the CLR timeout is less than the *Resetting the NAV* timeout. Assuming that the neighbouring nodes are stationary in terms of network topology, the recipients of the CLR packet will be the same as the recipients of the CTS packet. In this case, node **X** will receive a CLR(D,S) packet which matches the previously received CTS(D,S) packet. If this is the case, it will clear its NAV and be able to contend for access to the channel, knowing it will not cause a collision at node **D**. A similar technique is proposed by Du, et al. called the receiver initiated NAV clearing (RINC) scheme (Du & Chen 2005), and operates using CLR packets. In this thesis, the standard non-modified IEEE 802.11 MAC layer is used during simulation evaluations to allow comparison to other research results.

2.5.3 Deferring Mechanism

The deferring mechanism is initiated when a node has been unsuccessful at contending for access to the wireless channel. This is due to “busy” responses by

either the physical carrier sense provided by the PHY layer or the virtual carrier sense provided by the MAC layer. The unsuccessful node must defer contention until the channel has been idle for a minimum of a DIFS interval if the previous frame was received correctly, or an extended inter-frame space (EIFS) interval if the previous frame was corrupted. After either the DIFS or EIFS interval has elapsed, a random back-off scheme is employed to restrict repetitive contention by the same node for channel access. The random back-off scheme (also termed the binary exponential back-off scheme) provides an additional waiting interval, called the *back-off time*, for the node to endure before it is allowed to contend for channel access again. The random back-off time, B_t, is calculated using equation 2.7:

$$B_t = random() \times aSlotTime \qquad \text{(Eq. 2.7)}$$

where *random()* is a pseudo-random number generator that generates an integer between $[aCW_{min}, CW]$, where CW is bounded by $[aCW_{min}, aCW_{max}]$, and $aSlotTime$ is defined by the PHY layer characteristics given in Table 2.3.

Characteristic	Value
aSlotTime	20 μs
SIFS	10 μs
DIFS	50 μs
EIFS	1148 μs
aCW_{min}	31
aCW_{max}	1023

Table 2.3 – Direct-sequence spread spectrum (DSSS) physical layer characteristics for the IEEE 802.11b standard (IEEE Std 802.11 1999).

The contention window (CW) value is selected from a range of integers between aCW_{min} and aCW_{max}, and depends on the number of retransmissions attempted by a transmitting node. The contention window is given an initial value of aCW_{min} ($2^5 - 1 = 31$) and is exponentially increased after each unsuccessful transmission until a maximum value of aCW_{max} is reached ($2^{10} - 1 = 1023$), as shown in Figure 2.9.

Thus, the minimum random back-off time, calculated using equation 2.7, is 620 μs and the maximum random back-off time is 20.46 ms. After successful transmission by a node, the contention window value is reset to the initial value of aCW_{min} again.

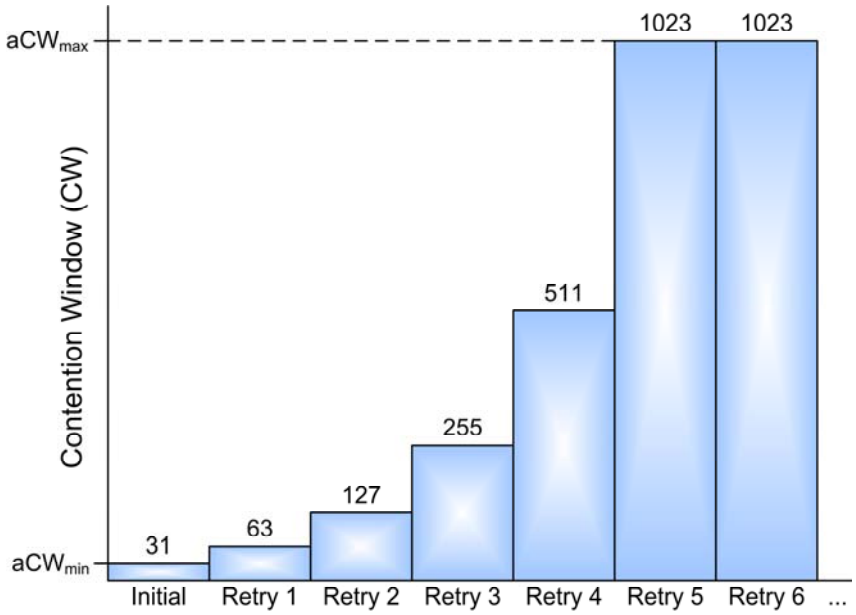


Figure 2.9 – The contention window (CW) value is selected from a range of integers between the initial value of aCW_{min} and a maximum value of aCW_{max}. The range of CW is exponentially increased for each unsuccessful transmission, until aCW_{max} is reached. This figure is adapted from the IEEE 802.11b standard (IEEE Std 802.11 1999).

Once a node has successfully contended for access to the channel, the CW is reset to aCW_{min}, and the RTS-CTS-DATA-ACK sequence is permitted to proceed, as discussed in Section 2.5.1. It should be noted that the source node wishing to send an RTS packet must wait for a DIFS interval before transmitting, given the physical carrier sense and NAV permit the transmission to take place. Following this, the source node and destination node only wait a SIFS interval between all other packet exchanges in order to hold onto the channel. This allows the data transmission sequence to complete without interruption by any surrounding nodes' RTS packets, which have to wait for a longer DIFS interval before channel access is allowed.

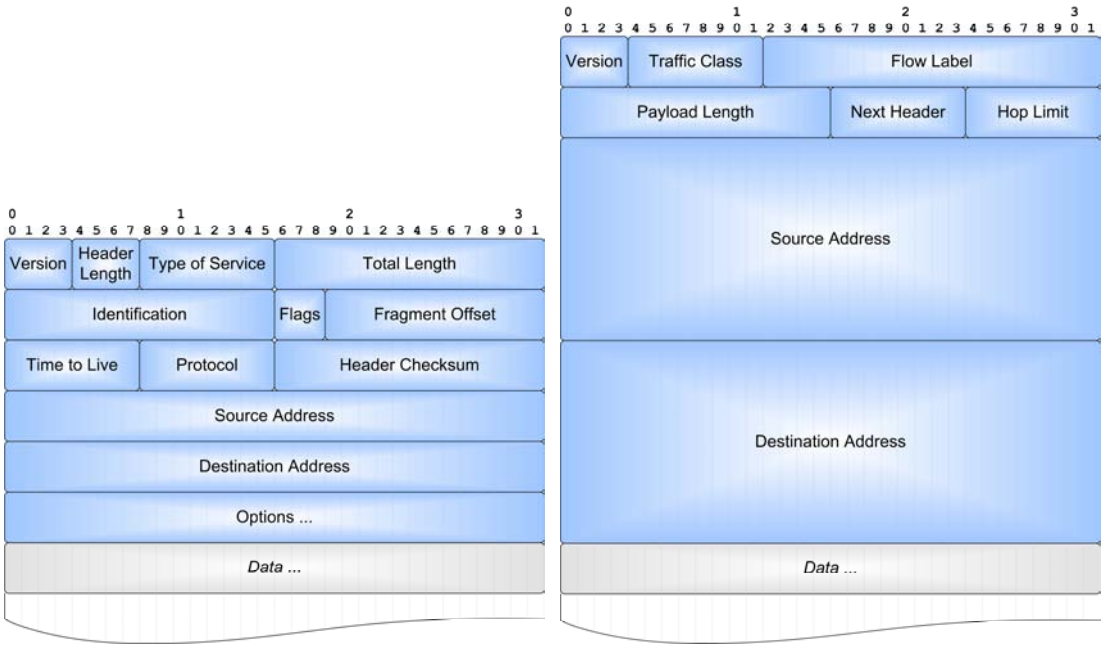
The IEEE 802.11 MAC layers' distributed coordination function (DCF) has two serious channel contention problems, one based on the exposed node problem, and

the other on the binary exponential back-off scheme (Xu & Saadawi 2001). The exposed node problem causes a condition known as the "TCP instability problem", whereby packet collisions will prevent a node from transmitting data to its next up-link node (Xu & Saadawi 2001). After retransmission attempts of the packet have failed, the MAC layer will cause a link-failure to be reported to the routing layer resulting in a broken routing path. The impact of this problem in terms of packet throughput and latency is evaluated in Section 7.3.7. The binary exponential back-off scheme causes a condition known as the "serious unfairness problem", in which the last node to successfully contend the channel is favoured in terms of reacquiring the channel in subsequent attempts (Xu & Saadawi 2001). This is because the node that successfully transmitted has its CW reset to aCWmin, whereas neighbouring nodes would have an exponentially higher CW value because of failing to successfully transmit on the channel. Overall, these two problems will have considerable impact in the design of the routing protocol, as it must deal with these two MAC layer related problems embedded within the IEEE 802.11 MAC layer.

2.6 NETWORK LAYER

The network (NET) layer is responsible for the routing of data traffic received from the higher layers, namely the transport layer and application layer, and for maintaining an accurate routing table of the network. The NET layer is defined on Layer 3 of the OSI model and Layer 2 of the TCP/IP stack. This layer is concerned with logical addressing and path discovery using protocols such as Internet Protocol (IP) and various routing protocols. The IP addresses are logical addresses, such that they are assigned by a person to a physical device, where each physical device inherently carries a unique physical MAC address. The mapping of IP addresses to MAC addresses is performed by the address resolution protocol (ARP) (Plummer 1982) which resides in this layer. When a node needs to resolve an IP address, the ARP protocol will consult its local ARP cache for the corresponding MAC address. If it cannot be resolved or if the entry within the ARP cache has timed out, the ARP protocol will broadcast an ARP request packet.

Above the ARP protocol lies the routing protocol which is required to provide the basic functionality of obtaining a valid routing path between any reachable source and destination node, usually in the form of a sequence of IP addresses. There exist two types of IP protocols in use today, IP version 4 (IPv4) and IP version 6 (IPv6). The 32-bit IPv4 (Postel 1981) was introduced in 1981 and provides up to 2^{32} (~4.3 billion) unique network addresses, which is insufficient for the future of the Internet beyond 2011 (IANA 2008). The replacement for IPv4 is the 128-bit IPv6 (Deering & Hinden 1998) which was introduced in 1998, and provides a pool of 2^{128} (~340.3-billion-billion-billion-billion) unique network addresses. The IPv4 packet structure consists of 13 fields, as shown in Figure 2.10(a), and the IPv6 packet structure consists of 8 fields, as shown in Figure 2.10(b).



(a) (b)

Figure 2.10 – IPv4 and IPv6 packet structures, shown in (a) and (b), respectively.

It should be noted that the IP protocol is the underlying protocol used in this thesis. An extensive review of routing protocols that are built upon the IP protocol foundation is provided in Chapters 3 and 4.

CHAPTER 3

REVIEW OF ROUTING PROTOCOLS

3.1 INTRODUCTION

The term “routing” refers to the process in which a path (or route) is selected between a source node and a destination node along which to transmit data. As discussed in Section 2.3, a routing protocol is defined in layer 3 of the OSI model and layer 2 of the TCP/IP stack, both of which are referred to as the network layer.

When routing is used in conjunction with circuit-switching, a dedicated connection is maintained for the entire duration of the transmission between the source and destination. Such a technique can be inefficient as valuable channel resources are consumed even if the channel is idle.

Packet-switched networks on the other hand, have primarily been used to carry information in discrete packets. In this case, data packets, possibly originated from different sources, are routed to follow various paths across the network to reach their destinations. The overall efficiency and effectiveness of packet switching is therefore greatly dependent on the routing protocol chosen.

3.2 OBJECTIVES OF A ROUTING PROTOCOL

The primary goal of any routing protocol is to discover a suitable path between any reachable source-destination pair to be identified in a timely manner. Often, an effective routing protocol is required to possess many other desirable characteristics, including the followings:

- Low latency for route discovery (request) and acquisition (reply).
- Provide an accurate representation of the current state of the network topology and connectivity between nodes.
- Adapt rapidly to changes in topology due to nodes joining and leaving the network.
- Select an optimal routing path based upon certain selection criteria or performance metrics, such as minimum hop-count, link usage and stability, bit-error-rate, bandwidth capacity, link latency/delay, length of queue, etc.
- Minimise the overhead of the routing protocol by reducing the update packet frequency, and size of the update packet.
- Offer multiple alternative and independently redundant (disjoint) routing paths as a means for fail-safe routing and congestion control.
- Low computational complexity in its implementation.

The fundamental key to the integration of these characteristics into a practical routing protocol is through the sharing of information between nodes. A number of distinct processes are employed to achieve the above characteristics, of which, information pertaining to the connectivity between nodes is the most vital. Such information is necessary for a given node to be able to identify the topological arrangement of the network. With this knowledge, any individual node is able to collect and store the operational statistics of the nodes within its awareness range.

A routing protocol is able to determine the best routing path between a given source node and its destination based on specified selection criteria and performance metrics. For example, using a combination of bit-error-rate (BER) and congestion statistics, of which the individual metrics can be independently weighted according to their particular level of importance. Moreover, the most commonly used routing metric is the hop-count, defined as the number of intermediate relay nodes a packet must traverse in order to reach its destination (Chen, Druschel & Subramanian 1999). It should be noted that in order to maintain congruency between calculated metrics, each node in the network should operate the same routing protocol.

Next, it is important for a routing protocol to be able to determine the optimal path as quickly as possible, i.e., fast convergence. When connections between nodes are created and destroyed, if it takes a long time to update each node, the network may become unstable in the sense of causing routing loops and unreachable destinations. Therefore, connectivity information must be accurately and timely distributed amongst the nodes with minimum delay.

Furthermore, a routing protocol has to be robust by being able to offer fail-safe and adaptive routing. When adverse and unexpected connectivity changes occur, it is expected that the routing protocol should continue to operate normally. For example, network failures could be avoided if alternative paths are available to repair or modify a broken routing path, as shown in Figure 3.1.

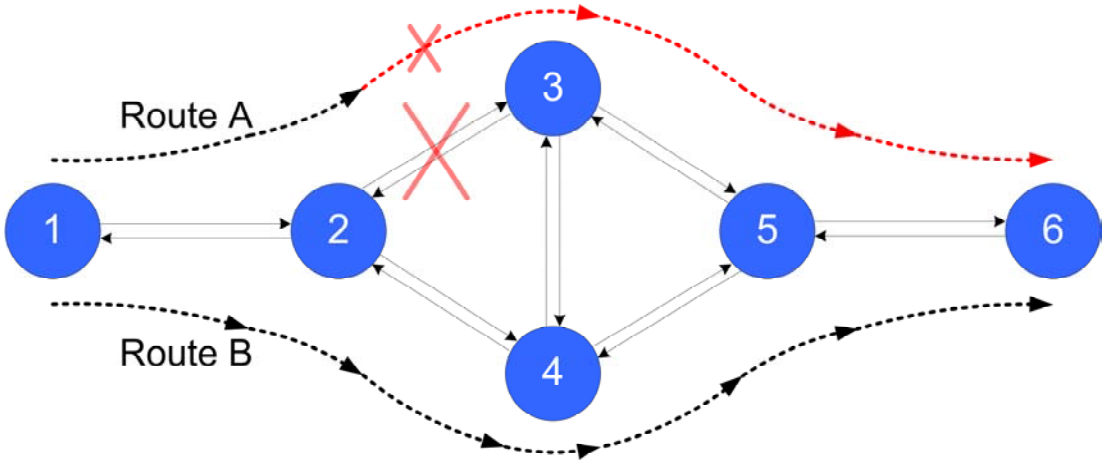


Figure 3.1 – Alternative routing paths as a means for fail-safe and adaptive routing. A link breakage for route **A** is shown (in red) between nodes **2** and **3**, therefore, node **2** could select an alternative unbroken route **B** (in black) via node **4**.

For practical consideration, preference is given to a routing protocol with low computational complexity. This attribute allows the routing protocol to operate efficiently on networks with limited computational and energy resources, e.g., an isolated rural wireless network.

3.3 ROUTING PROTOCOL FUNDAMENTALS

Most routing protocols are based on two general algorithms that were originally devised for wired networks; namely distance-vector routing, and link-state routing. These two algorithms differ in the method used to calculate and select the optimal routing path between a source and destination node. The concepts of these algorithms are now discussed with a brief overview of graph theory, emphasising on directed graphs.

3.3.1 Directed Graphs

A graph can be constructed by means of connecting edges and vertices together to form a topological representation of a network. In terms of a routing protocol, an edge can be viewed as a path connecting two nodes together, and each node is called a vertex. For a wireless ad-hoc network, many connections can be formed between individual nodes. As such, a graph can consist of many vertices with multiple connecting edges. Furthermore, wireless ad-hoc networks exhibit both unidirectional and bidirectional links, which are represented in a graph as a single edge or double directed edges, respectively. Within a graph, each edge is associated with a certain link cost that is used as a routing metric. Among the many different criteria normally associated with the link cost, the most common metric is the hop-count distance between each respective node (Chen, Druschel & Subramanian 1999). Other metrics on which route selection can be based include link usage, bit-error-rate, bandwidth capacity, link latency, and congestion statistics. For a graph with vertices connected by multiple bidirectional edges and associated link costs, it is mathematically referred to as a directed graph network with multiple edges, or in short, a directed multigraph network.

A directed graph, represented by $G = (V, E)$, can be expressed as a pair of disjoint sets of vertices and edges, together with two maps $init : E \to V$ and $ter : E \to V$, where V and E are the vertices and edges, respectively (Chartrand 1984; Diestel 2000). The two maps assign an initial vertex $init(e)$ and a terminating vertex $ter(e)$

to every edge e, which is represented by an ordered pair $\{i, j\}$ where $i = init(e)$ and $j = ter(e)$. For example, graph $G = (V, E)$ defined with vertices $V = \{1,2,3,4,5\}$ and edges $E = \{\{1,2\},\{1,3\},\{2,1\},\{2,4\},\{3,2\},\{3,4\},\{3,5\},\{4,2\},\{4,5\},\{5,3\}\}$ is shown in Figure 3.2.

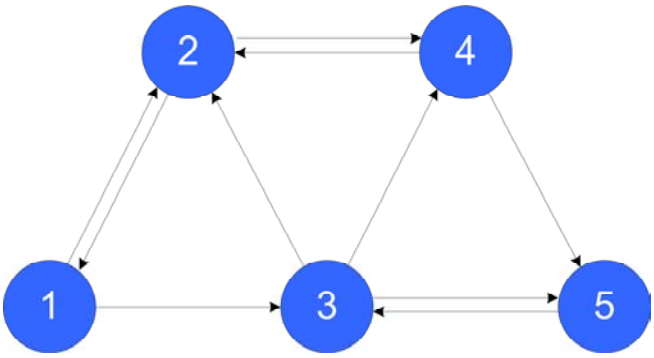
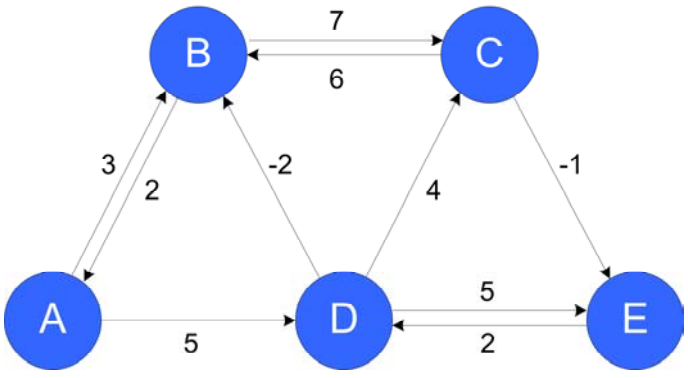


Figure 3.2 – A directed graph consisting of 5 vertices and 10 edges.

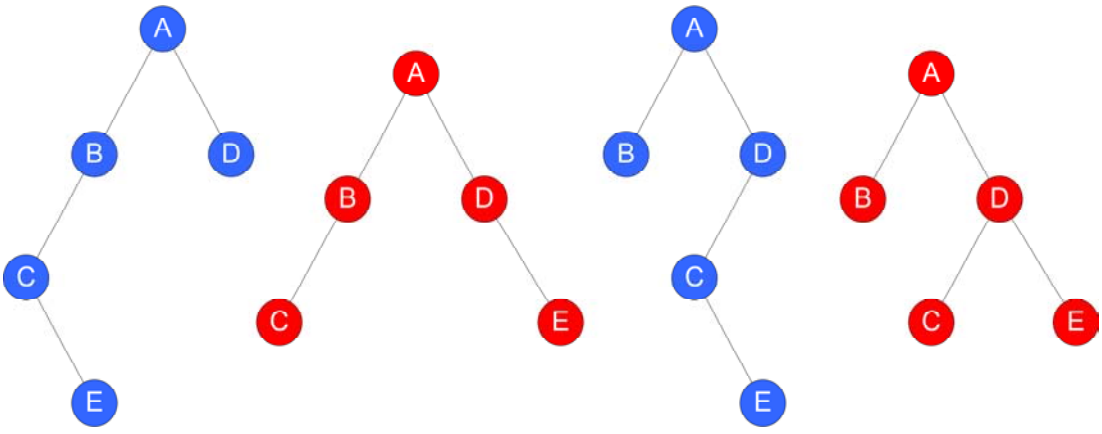
Similarly, a directed graph can also be represented by an $(n \times n)$ adjacency matrix of G, which is defined by $A = (a_{ij})_{n \times n}$, where $a_{ij} = \begin{cases} \alpha & \text{if } v_i v_j \in E \\ 0 & \text{otherwise} \end{cases}$, and v is the vertex, and α has a value equal to the link cost. Using the same directed graph in Figure 3.2 as an example, the equivalent adjacency matrix with artificial link costs is shown in Figure 3.3.



	A	B	C	D	E
A	0	3	0	5	0
B	2	0	0	7	0
C	0	–2	0	4	5
D	0	6	0	0	–1
E	0	0	2	0	0

Figure 3.3 – A directed graph consisting of 5 vertices and 10 edges showing its associated link cost adjacency matrix.

In a computational environment, the mathematical expression for a directed graph and its equivalent adjacency matrix can be also viewed as a tree structure. A tree structure consists of nodes that correspond to the vertices of the graph, and also of connections between the nodes which represent the edges of the graph. It is evident from the graph, shown in Figure 3.3, that multiple trees could be drawn to depict the graph due to the routing loops evident, and is illustrated in Figure 3.4(a–d).



(a) (b) (c) (d)

Figure 3.4 – Multiple tree combinations depicting a single directed graph. The four combinations shown in parts (a–d) originate at node **A** as the source node.

For this reason, to utilise a tree structure in a routing protocol, the spanning tree of the graph should first be found. As a consequence of finding the spanning tree, all routing loops are eliminated as vertices are not repeated. The spanning tree is not unique, as described by Kirchhoff's Matrix-Tree Theorem (Kirchhoff 1847). This theorem states that the number of unique spanning trees possible for a graph G is equal to any cofactor of the degree matrix of G minus the adjacency matrix of G. When link costs are not considered, e.g., an unweighted graph, any spanning tree of the graph G is also the minimum spanning tree. If link costs are considered, however, a minimum spanning tree is defined as a spanning tree that connects all n vertices in graph G using $n-1$ edges with minimum overall weight. The term minimum spanning tree is also called graph geodesic, and is synonymous with finding the shortest path with minimum distance $d_{\min}(u,v)$ between two vertices (u,v) of graph G, where u is the source vertex, and v is the destination vertex.

Four common algorithms for determining the minimum spanning tree are the breadth-first traversal algorithm (Preiss 1998), the reaching algorithm (Jensen & Bard 2002), the Bellman-Ford algorithm (Bellman 1958), and Dijkstra's algorithm (Dijkstra 1959). The latter two algorithms will be discussed in greater detail for the case of Distance-Vector and Link-State routing algorithms, respectively, in the following sections.

3.3.2 Distance-Vector Algorithm

A distance-vector routing algorithm operates by periodically transmitting an update packet to all of its neighbours through flooding. Flooding is an application of repetitive broadcasting by subsequent nodes, or by sequentially transmitting to each neighbouring node individually, termed unicasting. The update packet consists of two categories of reachability information regarding nodes, provided as follows:

- All neighbouring nodes which are directly reachable within 1 hop, and
- All other nodes (of which the source node is aware) further than 1 hop away.

As the title of the algorithm suggests, the information provided for each category within the update packet is two-fold. Firstly, the distance from itself to any other node that it is aware of, and secondly, the direction vector required to reach these nodes. The distance is measured in terms of a routing metric or cost function, and is usually the number of physical hops separating the current node to a specific target node. The direction vector required to reach this specific node is given in terms of the nearest neighbour who can reach it. In other words, the neighbouring node that is capable of getting 1 hop closer to the target destination node. This is often referred to as the next-hop or uplink node. A routing path is then selected based on the lowest overall cost from the current node to the target destination node, and the packet is subsequently forwarded to the corresponding uplink node.

A common algorithm to calculate the minimum cost routing path for a distance-vector based routing protocol is the Bellman-Ford algorithm. The operation of the algorithm can be best illustrated by means of a directed graph, adapted from Figure

3.3, showing the transitions between steps and the resulting routing table, as shown in Figure 3.5(a–h) and Figure 3.5(i), respectively.

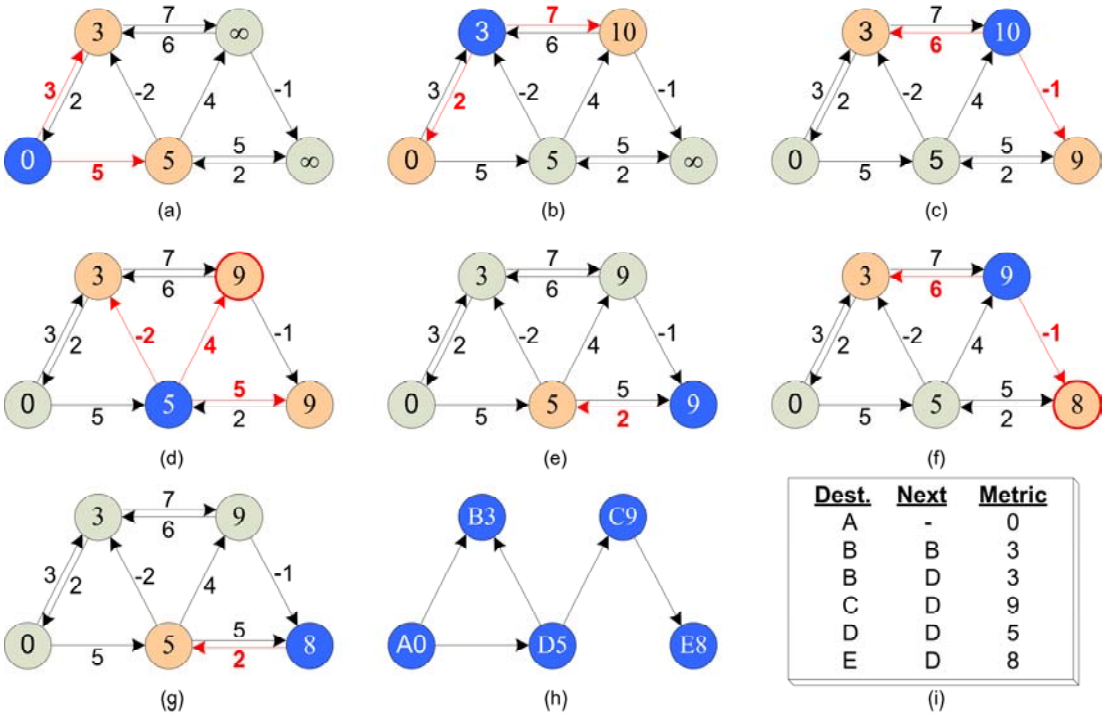


Figure 3.5 – Example of the Bellman-Ford algorithm to obtain the minimum cost spanning tree path. Parts (a–g) show the step-by-step transitions (shown in red) to obtain the minimum distance between each node (shown in blue) and its neighbours. The final minimum cost spanning tree is shown in part (h), with the corresponding routing table shown in part (i).

The cost function associated with traversing between two vertices is the weighting assigned to each edge, and this can be either a positive or negative value. It should be noted that the algorithm will breakdown if a negative-weight cycle occurs, e.g., when the sum of the cost of the forward and reverse traversals is less than zero, as given by $\delta(u_i, v_i) + \delta(v_i, u_i) < 0$.

The principle of the algorithm is straightforward, and operates according to the following three steps:

1. For all N vertices, let $d(v_1) = 0$, and $d(v_i) = \infty$, where v_1 is the source vertex, and v_i consists of the remaining nodes where $i = [2..N]$.
2. For every edge on each vertex, let $d(v_i) = d(u_i) + \delta(u_i, v_i)$ if $d(v_i) > d(u_i) + \delta(u_i, v_i)$, where $d(v_i)$ is the interim minimum distance traversing from u to v, given $d(u_i)$ is the interim minimum distance to u.
3. Verify that the negative-weight cycle, $\delta(u_i, v_i) + \delta(v_i, u_i) < 0$, has not yet occurred (as discussed above).

From a distance-vector routing protocol point of view, each iteration of the Bellman-Ford algorithm operates on the routing metrics provided by the update packets received from its neighbouring nodes. The routing metrics provided by the neighbouring nodes are their respective minimum distances to all other nodes within the topology, calculated using the same Bellman-Ford algorithm. Therefore, the routing table locally constructed at each node contains the minimum distance $d_{\min}(u, v)$ from u to v as the routing metric, where u is the current source vertex, and v is the destination vertex. The uplink node v_i is therefore the corresponding neighbour of u that can reach v.

The disadvantage of the Bellman-Ford algorithm is that it is not suitable for large networks as it does not scale well. This is due to an implication of the "counting-to-infinity" problem. The counting-to-infinity problem can be illustrated using the following example, and is shown in Figure 3.6.

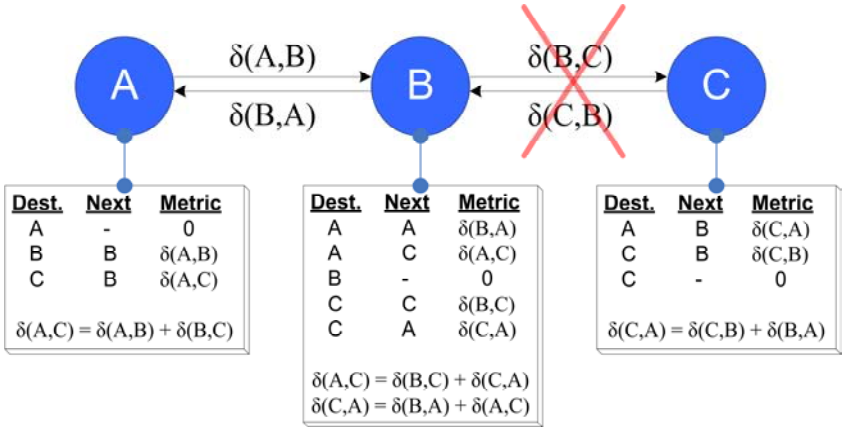


Figure 3.6 – Counting-to-infinity problem in the Bellman-Ford algorithm for three nodes, showing their respective routing tables.

Assume node **A** is dependent on its uplink node **B** to reach a particular destination node **C**. When uplink node **B** loses reachability with the destination node **C**, a counting-to-infinity problem occurs. This is when uplink node **B** now assumes node **C** is reachable via node **A**, due to node **A**'s announcement that it can reach node **C**. Since the routing path is not explicitly made known, as in source-routing, node **B** has no way of knowing that node **A** uses node **B** as its uplink to reach node **C**. Therefore, the situation arises where node **B** will add its cost metric $\delta(B,A)$ to $d_{\min}(B,C)$ in order to use node **A** as its uplink, which in turn causes node **A** to add $\delta(A,B)$ to $d_{\min}(A,C)$. From this, it is apparent that both node **A** and **B** will repeatedly continue to add $\delta(B,A)$ and $\delta(A,B)$ until the minimum distance to node **C** tends to infinity. For this reason, to inhibit the counting-to-infinity problem, infinity is defined as a relatively small number, namely 16. The major implication is that this causes the routing protocol to not scale beyond 15 possible hops, limiting its network topology size. The most common distance-vector routing protocol is the Routing Information Protocol (RIP) (Hedrick 1988), which is discussed in Section 3.5.1, and is based on the Bellman-Ford algorithm.

3.3.3 Link-State Algorithm

A link-state routing protocol reduces the amount of broadcast traffic by not transmitting periodic routing update packets as per distance-vector protocols. Instead of transmitting the entire routing table each period, a link-state routing protocol only transmits the entire routing table once, e.g., upon neighbouring node initialisation. Thereafter, the protocol only transmits the connectivity changes that have occurred in the topology to its neighbouring nodes in the form of a link-state advertisement (LSA). The LSA is broadcast by method of flooding as soon as the link-state of a node changes, in other words, when a node becomes active or inactive the LSA packet will be transmitted immediately. As a result, each node receiving the LSA packet will independently recalculate all the routing paths accordingly and maintain a topological map of the network. A routing path is then selected based on the lowest overall cost from the current node to the target destination node, as per the distance-vector protocol.

The most common algorithm to calculate the minimum cost routing path for a link-state algorithm is Dijkstra's algorithm. Since every node has a topological map of the entire network available, each node can represent this information in terms of a directed graph. Dijkstra's algorithm is then used to calculate the distances between any given vertex u and all other vertices v_i in the graph, where u is the current source vertex. It should be noted that unlike the Bellman-Ford algorithm, Dijkstra's algorithm must have non-negative edge weightings. However, the advantage of Dijkstra's algorithm is the ability to handle cyclic loops (counting-to-infinity problem) within the graph, provided it is not a negative-weight cycle (Preiss 1998). The algorithm operates according to the following three steps:

1. For all N vertices, let $k_{v_1} = 0$, and $k_{v_i} = \infty$, where v_1 is the source vertex, and v_i consists of the remaining nodes where $i = [2..N]$.
2. For the vertex with the smallest k_{v_1}, select the adjacent edge with the lowest weight to v_i.
3. Add the lowest edge weight to $k_{v_2} = k_{v_2} + \delta(u_i, v_i)$.

The first step taken is to initialise the source vertex with a value of $k_{v_1} = 0$, and all other vertices with value $k_{v_i} = \infty$, where i is the i^{th} vertex and k_{v_i} is the interim cost to reach vertex v_i from v_1. The algorithm takes N iterations to find the distance between the source vertex v_1 and all other vertices v_i, where N is the number of vertices in the graph. Then, the vertex with the lowest k_i value is selected for the first iteration of the algorithm, which initially would be k_{v_1} since it is equal to zero.

The next step taken is to select the minimum cost edge connecting the current vertex with an adjacent vertex. The value of k_i for the adjacent vertex is then set to the current vertex cost plus the cost of the edge connecting the two vertices. For all $N-1$ subsequent iterations, the vertex with the lowest k_i value is selected each time excluding the previously covered vertices. This continues until eventually the minimum spanning tree is found. The sequence in which the paths are found are in

the order of the summation of minimum edge weights, i.e., the shortest routing path is found first, and the longest path is found last.

The operation of the algorithm can be best illustrated by means of a directed graph, adapted from Figure 3.3, showing the transitions between steps and the resulting routing table, as shown in Figure 3.7(a–e) and Figure 3.7(f), respectively. By observing the routing table obtained in Figure 3.7(f), the minimum weight routing path from node **A** to node **E** is determined to be **A-D-C-E**, and not **A-D-E**, even though the overall weight of 10 is the same. This is because weight 4 is selected before weight 5 at node **D** in the step shown in Figure 3.7(c), regardless of the fact that node **C** must add 1 to reach node **E**, yielding $4+1=5$. Nevertheless, if the edge weightings were to refer to hop-count, the path **A-D-E** would obviously be shorter than **A-D-C-E**. Therefore, after N iterations all the shortest paths will be known when the minimum weight refers to hop-count. It should be noted that the routing table produced using the Bellman-Ford algorithm in Figure 3.5(i) is different from Figure 3.7(f) which uses Dijkstra's algorithm, by observing that the graph has non-negative link weights.

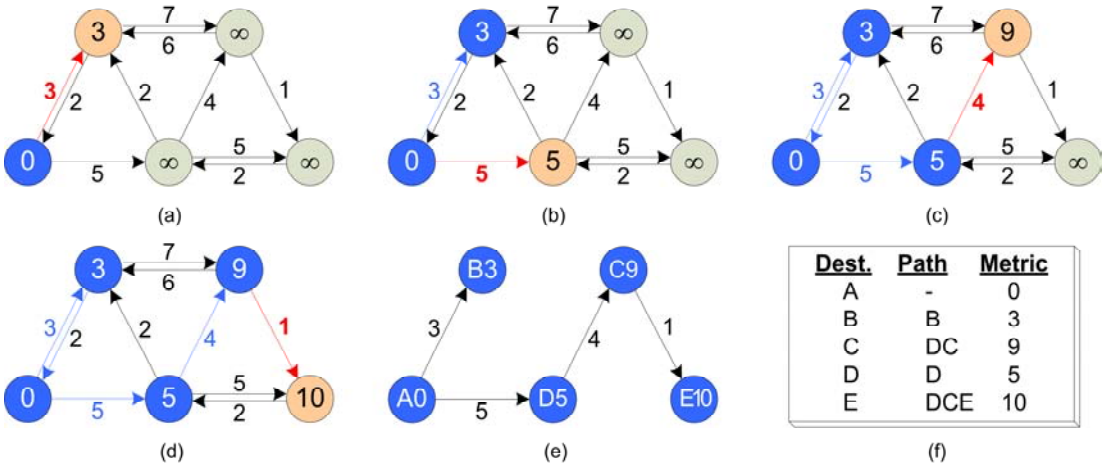


Figure 3.7 – Example of Dijkstra's algorithm to obtain the minimum cost spanning tree. Parts (a–d) show the step-by-step transitions (shown in red) to obtain the minimum distance between each node (shown in blue) and its neighbours. The final minimum cost spanning tree is shown in part (e) with the corresponding routing table shown in part (f).

3.4 ROUTING PROTOCOL ORGANISATION

The selection of a routing protocol is highly dependent on the type of network in which it is being deployed (Ballew 1997). The organisational structure of a network determines which device is responsible for the routing of packets between nodes within a network. As such, routing protocols can be broadly classified into two distinctive models with two schemes per model; namely, centralised or decentralised, and static or dynamic routing.

3.4.1 Centralised Routing

A centralised routing scheme is when a single node within a network is responsible for managing all routing processes. This single node must have a global topological view of the entire network, and is therefore most suitable for fixed infrastructure-based networks where the connectivity between nodes changes infrequently. The major advantage of using a centralised routing model is the ability to pre-compute and pre-determine the optimal routing paths throughout the network. This also allows for the removal and restoration of routing paths during network modifications.

The major disadvantage of a centralised scheme is the inability of any single node to cope with the routing load after a sizeable network failure has occurred, for example, power failure. At this point, the centralised router would be required to rebuild the global routing table for the entire network, and this would take a considerable amount of time for any large network. During this rebuilding time, any node that makes a request for a valid routing path between a source-destination pair would be unanswered by the centralised router, which would likely be congested with such requests. A single-hop wireless ad-hoc network is an example of such a system utilising a centralised routing scheme. As such, the access point (AP) must act as a centralised router that handles all routing functionality for the subnet formed by the STAs connected to it.

3.4.2 Decentralised Routing

A decentralised routing scheme is where the burden of routing functionality is distributed among all the nodes in the network. As a consequence, each node is therefore responsible for computing its own routing table. In this situation, there is no single node taking global responsibility for routing information, as is the case for centralised routing. As a result, each node must execute its own routing algorithms. An example of such a system using a decentralised routing scheme is that of a multi-hop wireless ad-hoc network, as previously mentioned, in which each STA is also a routing device.

The major advantage of a decentralised system is its immunity to global network failure when compared to a centralised scheme. However, this comes at the cost of higher routing traffic overhead since all nodes within the network must maintain their own routing tables instead of relying on a centralised routing table. The increased traffic overhead comes from each node transmitting its routing table knowledge to its neighbouring nodes, as will be described in further detail in Section 3.4.4. Given the increased traffic, it should be pointed out that it becomes an enormous burden and a cumbersome task to maintain a global topological view of the entire network for each node within the network. Later in Chapter 5, a method will be described on how to combat this problem.

3.4.3 Static Routing

The approach by which a node becomes aware of its surroundings, i.e., the network topology, determines whether a routing protocol is static or dynamic in nature. A static protocol is defined by a routing table that is pre-defined on a node-by-node basis for the entire network. This type of protocol is predominantly suited to networks that do not change over time, or if they do, only very infrequently. A typical example of such a network is the deployment of two fixed infrastructure-based local area networks (LANs). This is where internet protocol (IP) addresses are statically assigned to each node, and traffic is forwarded to the respective default gateways on each subnet using static IP addresses.

There are several advantages of having a static routing scheme, of which route predictability can be considered a primary benefit. Unless the routing table is manually reconfigured, there is no routing overhead incurred and the latency experienced by a packet as it traverses the network is effectively stable and fixed as the routing path remains constant. Another benefit is that a static routing scheme is relatively straightforward to configure and maintain for small networks, however, this benefit diminishes as the scale of the network size increases.

A static routing scheme also has its disadvantages. At the outset, if any node in the network fails, it becomes impossible to circumvent the failed node by utilising another routing path, due to the selection of the routing path being fixed. Even though potentially alternative paths are available, in the case if a node has more than one uplink gateway, the link cannot be utilised unless a backup route is statically pre-defined within the routing table. As previously mentioned, the static routing scheme leads to scalability problems and is generally unsuitable for larger networks. The scalability issue arises from the fact that any single change in the network topology must be reflected and manually changed in the routing tables of all the nodes within the network which can be time consuming.

3.4.4 Dynamic Routing

Dynamic protocols, on the other hand, do not suffer from scalability problems as with static protocols. Dynamic protocols allow adaptive changes in the topology to take place by continuously monitoring the state of the neighbouring nodes and updating the routing tables accordingly. This allows for the network to grow and change over time, but also allows for network failures such as link failures to be corrected as they occur. If a change in the network topology occurs, such as a node becoming active or inactive, the node that detected the change in the topology will transmit a routing update packet. In the case when a node failure has occurred, it would simply appear as a node becoming inactive to its neighbouring nodes. As such, the node that failed would be removed from the routing table by its neighbouring nodes, and possible alternative routing paths can use utilised to circumvent the failed node.

Dynamic routing protocols are therefore well suited to ad-hoc wireless networks which do not have fixed infrastructure-like network topologies; instead their topologies are allowed to change rapidly over time. The major drawback in dynamic routing is two-fold, and can be described as a complexity issue and a routing overhead issue. Firstly, the ability of a dynamic routing protocol to circumvent network failures by routing around the problem requires extensive and complicated algorithms to be designed. This causes the complexity of the routing protocol to be increased in terms of computational resources required by each node. Secondly, routing overhead is dramatically increased whenever a topological change occurs in the network, in the sense that every node must be made aware of the change that took place. As the routing table must accurately represent the network topology at any given time, each node must frequently update its neighbouring nodes about the connectivity status pertaining to its neighbours to maintain routing table integrity. Therefore, the sharing of correct and up-to-date information between nodes becomes most vital, as it allows nodes to select the most accurate routing path available upon request. As mentioned previously, the burden of increased traffic overhead and a method to combat this problem will be discussed in Chapter 5. Also, later in Chapter 4, two sub-types of dynamic routing will be discussed; namely, proactive and reactive routing protocols.

3.5 ROUTING PROTOCOL CLASSES

The topological construction of a network defines what class of routing protocols operates within the network. The classification is based on the standpoint placed upon the access point (or router) in terms of providing routing services to nodes within an “autonomous system”, defined hereafter. The routing protocols are divided into four main classes; namely, an Interior Gateway Protocol (IGP) (Hedrick 1988), an Exterior Gateway Protocol (EGP) (Mills 1984), a Border Gateway Protocol (BGP) (Lougheed 1990), or an ad-hoc network routing protocol.

An autonomous system is defined by RFC1930 / BCP6 (Hawkinson & Bates 1996) as “a set of routers under a single technical administration using an IGP with common metrics to route packets within the autonomous system, and using an EGP

to route packet to other (external) autonomous systems". This definition has since been relaxed to allow for multiple IGPs and metrics to be used within a single autonomous system. Overall, the organisation of an autonomous system must appear as having "a single coherent interior routing policy, and present a consistent…" and clearly defined representation of the nodes and "…networks reachable through it". The IGP, EGP and BGP protocols are discussed in greater detail in Sections 3.5.1, 3.5.2, and 3.5.3, respectively. An example network consisting of two autonomous systems utilising an IGP and EGP/BGP routing protocol is shown in Figure 3.8.

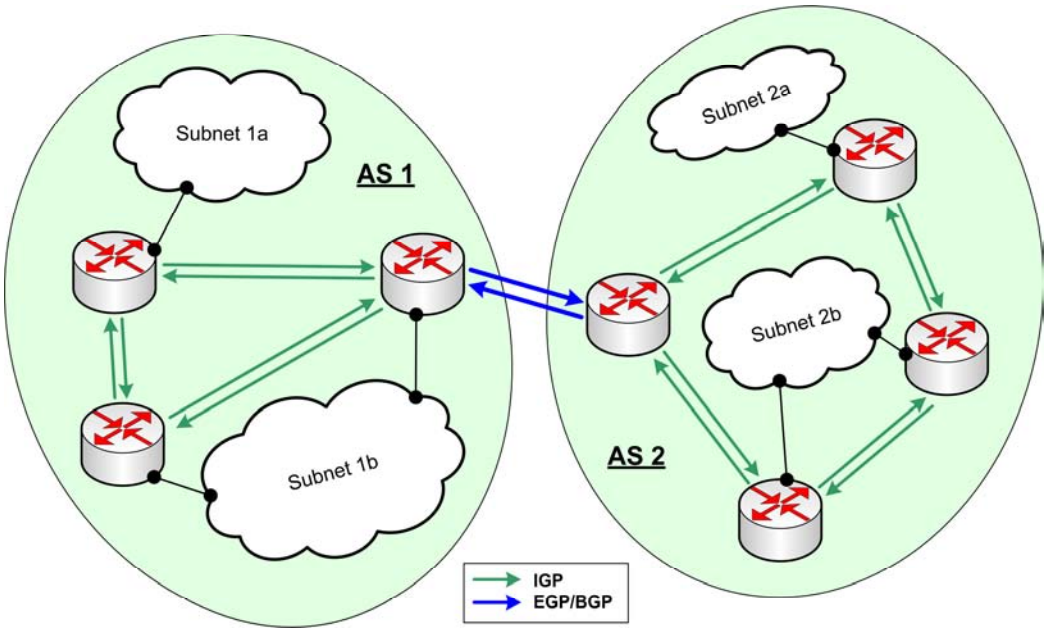


Figure 3.8 – Two autonomous systems, AS1 and AS2, showing IGP and EGP/BGP routers connecting the two subnets.

Lastly, an ad-hoc network routing protocol is not bound by the autonomous system definition. As such, it can be based on a loose amalgamation of both interior and exterior routing protocol characteristics, and is discussed in detail in Chapter 4.

3.5.1 Interior Gateway Protocol (IGP)

The routing protocol operating within an autonomous system is referred to as an interior gateway protocol (IGP). Three frequently used IGP routing protocols are the Routing Information Protocol (RIP) (Hedrick 1988), Interior Gateway Routing

Protocol (IGRP) (Cisco Systems 2005b), and the Enhanced Interior Gateway Routing Protocol (EIGRP) (Cisco Systems 2005a).

The RIP protocol is a distance-vector routing protocol, as discussed in Section 3.3.2, and uses hop-count as its primary metric to determine a suitable routing path. It operates by transmitting its routing table in a routing update packet periodically to other nodes within the network. The major shortcoming of RIP is that the protocol is limited to a maximum network size of 15 in terms of hop-count distance. This restriction severely limits the application of RIP to ad-hoc networks, and therefore, RIP would not be suitable for use in large networks due to the relatively low limit on the hop-count distance.

The RIP protocol was essentially superseded by IGRP which offered additional provisions for supporting a combination of multiple weighted routing metrics. Furthermore, it also increased the maximum hop-count from 15 to 255, effectively supporting a 17-fold larger network. IGRP was superseded by EIGRP which offered the benefits of link-state routing within a distance-vector framework. This allowed for faster convergence times, whilst at the same time reducing routing overhead by transmitting only connectivity changes instead of the entire routing table in each update packet. The EIGRP protocol is backwards compatible with IGRP, as well as supporting large-scale networks with the guarantee of loop-free routing. This functionality is achieved through exploiting the Diffusing Update Algorithm (DUAL) (Garcia-Luna-Aceves 1993) which enables the protocol to determine whether or not a path contains routing loops.

These three IGP protocols (namely, RIP, IGRP, and EIGRP) are near-obsolete in terms of being technically overtaken by two prominent link-state routing protocols. These being the Open Shortest Path First (OSPF) protocol (Moy 1989), and the Intermediate System to Intermediate System (IS-IS) protocol (Callon 1990). As discussed in Section 3.3.3, the link-state algorithm is more reliable in terms of determining global network connectivity. Also, it outperforms other distance-vector based protocols mentioned previously in terms of routing traffic overhead. This is because the update packet, termed the link-state advertisement (LSA), is quite small,

in the order of a few bytes, as compared to several hundreds of bytes usually needed for distance-vector based algorithms.

For OSPF and IS-IS, the LSA serves as an announcement packet, also called a "hello" packet, which is broadcast to all other nodes to inform them that the router in question is still operational. The routing path can then be determined from the topology constructed using the information contained within the LSA packets and the previously constructed routing table at the node. The routing protocol then calculates the minimum spanning tree with itself centred as the root. The advantages over distance-vector based protocols are at the expense of computational complexity, such that a link-state algorithm requires more processing to be performed at each router node.

The most prevalent IGP protocol is the OSPF routing protocol which supports multiple routing metrics and IP subnet masking (Thomas 2003). It should be noted that OSPF has been replaced by OSPF version 2 (OSPF-2) (Moy 1998) and more recently by OSPF version 3 (OSPF-3) (Coltun, Ferguson & Moy 1999) to support the new IPv6 addressing structure, as discussed in Section 2.6. The IS-IS protocol is technically similar to OSPF, except that IS-IS uses a different method in communicating between routers. This is accomplished by having dual router levels; one level for intra-subnet routing and the other for inter-subnet routing. As such, only routers sharing an equal level can share routing information directly. Furthermore, IS-IS is not natively an IP protocol and consequently never achieved widespread use as the de-facto IGP protocol for IP based networks.

3.5.2 Exterior Gateway Protocol (EGP)

The routing protocol used to interconnect two or more autonomous systems together is referred to as an exterior gateway protocol. The type of information passed between autonomous systems is called reachability information (Hunt 1997), which contains information regarding the networks that are reachable through a specific autonomous system. The first such protocol went by the same name as the classification, namely the Exterior Gateway Protocol (EGP), and was initially used to

interconnect the centrally controlled Internet backbone. The protocol operates by determining any neighbouring EGP routers through broadcasting a neighbour acquisition request. Upon reception of this request by another EGP router, it responds with either a neighbour acquisition acknowledgement confirmation or a neighbour acquisition refusal response. After this, it maintains the relationship between discovered neighbours using periodic "hello" and respective "i-heard-you" packets to keep the connection active.

After neighbour discovery, another periodic request packet is transmitted every 120-480 seconds to obtain the routing table contents from each EGP node, including the routing metrics used by the IGP. However, the details relating to the IGP routing metrics are not evaluated by the EGP. This is due to the inability to compare metrics between different IGP protocols that might be used within an autonomous system. Nevertheless, EGP remains a distance-vector based protocol even though it does not determine the optimal routing path. Instead, the IGP protocol assumes the responsibility of determining the optimal routing path.

There are two major shortcomings with EGP; firstly, the fact that the protocol is hierarchical, i.e., it is centrally controlled which means that scalability becomes an issue for large networks. Secondly, the protocol is not designed to guarantee loop-free routing. Routing loops can occur when the hierarchical tree of each autonomous system does not have a distinct common root node, as discussed in Section 3.3.1. An example of this is when a local area network is connected to the Internet using two different Internet gateways, and as such, duplication of nodes between each EGP can cause routing loops to occur during the interconnection process (Kozierok 2005).

3.5.3 Border Gateway Protocol (BGP)

The shortcomings of EGP led to the development of the Border Gateway Protocol (BGP) (Lougheed 1990). The BGP protocol allows for the creation of non-hierarchical network topologies that are not centrally controlled. The BGP protocol operates by periodically transmitting an "open" packet (similar to a "hello" packet) every 30-60 seconds to neighbouring BGP routers. In addition to this, the BGP

protocol will transmit a routing update packet containing only routing table changes to other neighbouring BGP routers when the router has detected a change in the topology.

One particular feature of BGP is that routing paths are deterministic, in the sense that a packet will traverse the same path whether or not the packet is originated at a certain node or simply passes through that node. For this reason, BGP is termed a path-vector protocol (also known as source-routing) since the routing path is entirely specified from the source node to the destination node. This is made possible because BGP accumulates the global routing tree that spans over multiple autonomous systems. Furthermore, the protocol has intrinsic loop-free routing due to its non-hierarchical nature, and therefore provides scalability to accommodate large networks. The characteristic of loop-free routing is a consequence of source-routing as BGP maintains a detailed knowledge of all the nodes within the autonomous system through which packets must traverse. As a result, BGP is able to detect routing loops and remove them from the routing table as they occur.

When compared to EGP, the BGP implementation supports multiple routing metrics. This allows multiple autonomous systems to be weighted according to a different cost function, and thereby allowing the route selection algorithm to choose the routing path of lowest cost. The most prevalent BGP implementation used today is BGP-4 (Rekhter, Li & Hares 2006), which replaces the now obsolete EGP protocol (Kozierok 2005).

3.6 SUMMARY

In summary, it is established that all routing protocols share the common objective of attempting to represent a changing network topology accurately in order to provide connectivity between nodes. The connectivity status between nodes must then be appropriately adapted given that nodes may enter or leave a network at any time, and this must be provided in a timely and efficient manner.

Descriptions of several routing protocol fundamentals have been examined, including the concepts of graph theory as it applies specifically to routing protocols. A number of fundamental routing algorithms were presented and illustrated, such as the distance-vector based approach using the Bellman-Ford algorithm, and the link-state based approach using Dijkstra's algorithm.

Furthermore, it is stated that routing protocols can be broadly classified according to their organisational structure, namely centralised or decentralised, and static or dynamic routing, and that these factors determine the behaviour of the routing protocol. Lastly, different routing protocol classes have been outlined, including the interior, exterior, and border gateway protocols for fixed networks, and how they apply to providing routing services to nodes within an autonomous system. This guides the discussion for ad-hoc network routing protocols, discussed next in Chapter 4, which utilises the knowledge gained from fixed network routing protocols.

CHAPTER 4

AD-HOC ROUTING PROTOCOLS

4.1 INTRODUCTION

Many ad-hoc routing protocols have been proposed for multi-hop wireless networks in the literature, and will be discussed in greater detail in Section 4.3. These protocols can be classified into several categories that define the type of the routing configuration they represent, and include:

- Flat routing (mesh-based networks),
- Hierarchical routing (cluster-based networks), e.g., ZRP (Haas 1997),
- Geographical routing, e.g., GZRP (Boukerche & Rogers 2001),
- Multicast/Geocast routing, e.g., MZRP (Zhang & Jacob 2003b), and
- Power-aware routing.

In this chapter, the focus will be on flat routing, and to a lesser extent on hierarchical routing. This is because although clustering allows for a reduced routing overhead in principle, a high-mobility ad-hoc wireless network can suffer due to the frequent cluster changes using hierarchical routing. This tends to result in unnecessary routing overhead as nodes move between clusters (Garcia-Luna-Aceves & Spohn 1999).

Therefore, the flat routing approach, whereby each node within the network has no hierarchical structure and uses a flat addressing scheme, is the method used within this thesis. Geographical based and power-aware routing protocols will not be considered in this thesis. The concepts of ad-hoc routing protocols are presented by means of a brief overview of the architecture of an ad-hoc network.

4.2 AD-HOC NETWORK OVERVIEW

Ad-hoc networks have no predetermined topological structure in the sense that interior gateway protocols (IGPs) or exterior gateway protocols (EGPs) encompass. As such, no distinct autonomous system, as detailed in Section 3.5, can be defined within such networks. Instead, a typical ad-hoc network consists of a system of "loosely connected" wireless devices each capable of routing and are thus not dependent on any infrastructure-based services for the network to operate. The term loosely connected refers to nodes being able to collaborate sporadically between one another for irregular periods of time. This can occur frequently because nodes are able to move freely in and out of transmission range of one another over time. An ad-hoc routing protocol must therefore be able to dynamically establish routing, and adapt to frequent topological changes in order to maintain reachability between the nodes comprising the network.

The simplest ad-hoc network consists of two nodes within direct transmission range of each other. Nodes that are not in direct transmission range form so called multi-hop networks that consist of three or more nodes, in which communication between nodes can be established through utilising intermediate relay nodes, as shown in Figure 4.1. In this example, nodes $\{A, B\}$ and $\{B, C\}$ are bi-directionally connected, as illustrated by the overlap in transmission range circles, but nodes $\{A, C\}$ are not connected since their respective transmission range circles do not overlap both nodes. For this reason, both nodes **A** and **C** must employ a route discovery procedure to discover node **B** in order to utilise it as an intermediate relay node.

When a data packet without a valid routing path in its header is presented to the medium access control (MAC) layer of a node, the node requests the routing protocol to provide it with a valid routing path between the source-destination pair. This routing path information is contained within the routing table that is stored and maintained by the routing protocol. The actions taken by a routing protocol to acquire the routing table information determines whether or not it is a proactive or a reactive protocol, and its operation is influenced by the design of the ad-hoc network.

In general, ad-hoc routing protocols obtain feedback regarding link breakage information from the MAC layer of the individual nodes. For example, in IEEE 802.11 devices, when the transmission retry counter is exceeded after multiple CTS or ACK timeouts, the MAC layer indicates to the NET layer that the uplink node has become unreachable. The routing protocol should then subsequently alter the routing table of a given node to reflect the removal of the uplink node in question.

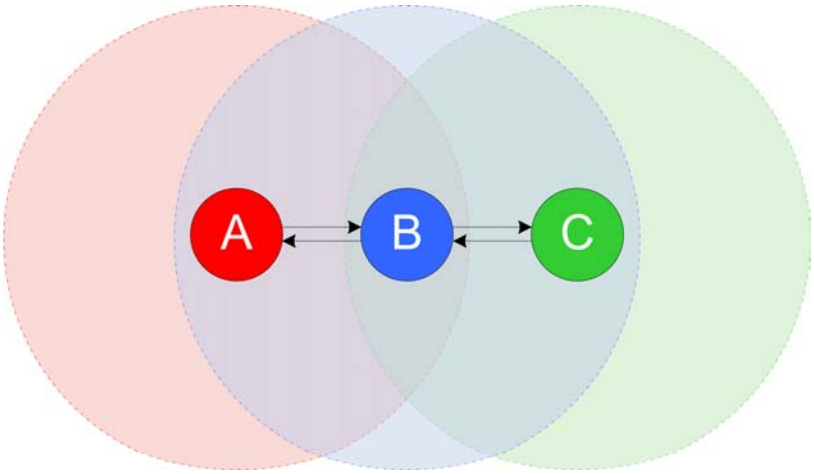


Figure 4.1 – A simple multi-hop ad-hoc network consisting of 3 nodes showing their respective transmission range circles.

Ad-hoc wireless communications is vastly different from the conventional wired communication networks, thus ad-hoc routing protocols must meet different design criteria than their wired routing protocol counterparts. Wired networks are able to support a large number of nodes whilst connectivity remains relatively unchanged over time, for example, a wide area network (WAN). Conversely, wireless ad-hoc networks usually have few nodes within transmission range of one another, of which the connectivity status changes frequently over time. As such, the general routing protocol principles that apply to wired networks, as described in Section 3.4, must be modified to meet the challenges faced by a wireless ad-hoc network. The two primary challenges faced by a wireless ad-hoc network are:

- To handle varying bandwidth capacities between nodes with respect to unpredictable time-varying channel conditions.
- To support frequent topological changes due to the mobility of nodes.

These two factors can be regarded as conditionally dependent on one another. Harsh channel conditions or low receive power results in the bandwidth throughput rate being progressively reduced in order to maintain communication between nodes. For instance, in IEEE 802.11b devices, a reduction from 11 to 5.5 Mbps or 2 to 1 Mbps can be encountered in order to maintain connectivity between nodes as the received signal strength becomes progressively lower (IEEE Std 802.11 1999). This scenario can occur when a node moves towards the edge of the transmission range or due to obstructions. This can be related to frequent topological changes in such a way that when the channel conditions are unable to sustain the minimum rate of 1 Mbps, the nodes will appear as having lost connectivity between each other. Therefore, wireless ad-hoc routing protocols must be designed in such a way as to effectively support both high and low bandwidth channels, as well as be able to adapt to frequent topological changes due to mobility.

4.3 TYPES OF AD-HOC ROUTING PROTOCOLS

There exist two distinctive strategies to address the above challenges described in Section 4.2; namely, proactive (active) routing, and reactive (passive) routing. Proactive routing achieves a trade-off in terms of having immediate routing information available at the expense of higher bandwidth utilisation due to periodic transmission of routing table information packets. Reactive routing, on the other hand, achieves a reduction of routing bandwidth utilisation at the expense of delays incurred to discover routes as needed on demand. For example, when a routing path is not immediately available within the routing table, a source node must first deploy a route discovery procedure in order to discover and acquire a valid routing path before being able to transmit a data packet. The time that elapses during this route discovery period consequently causes a reduction in throughput at the source node, and this is discussed in Section 4.3.2. Furthermore, proactive and reactive routing can be combined to produce an amalgamation of both proactive and reactive routing called hybrid routing (Cheng & Cao 2008), and is discussed later in Section 4.3.3.

4.3.1 Proactive Routing

A proactive routing protocol periodically transmits a routing information packet through broadcasting, much like a beacon. Therefore, an idle network, in the sense of carrying no data traffic, will still carry routing traffic for each node periodically. This is done in order to maintain the integrity of the routing table throughout the network, regardless of whether or not a connectivity change in the network topology has occurred. Periodic updating ensures that when a routing path is needed, no delay will be incurred as all the routing paths are already available in the routing table, provided that the destination node is reachable. As a side-effect, the routing table can contain routes that could potentially never be utilised.

As discussed in Section 3.4.4, it can be established that a proactive routing protocol also exhibits dynamic behaviour. As such, whenever a change in the topology occurs, the node that detected the change will generate a routing update packet. The routing update packet is then disseminated to its adjacent neighbours, usually through a method of flooding. This process continues until eventually all the nodes in the network become aware of the change in topology or connectivity information. From this definition, it is apparent that for a large network consisting of several hundred nodes, any change in the connectivity between nodes will produce an immense amount of routing traffic to be generated.

The scenario of redundant flooding is referred to as the “broadcast storm problem”, and is caused by transmitting excessive amounts of proactive routing traffic which is detrimental to network throughput. This routing overhead is detrimental to the efficiency of the network in terms of increased channel contention, and increased channel collision rates as examined by Ni, et al. (Ni et al. 1999). In this work, the author proposes probabilistic and counter-based improvements over the flooding approach to eliminate redundant transmissions. This is achieved by inhibiting some nodes from rebroadcasting for a random period of time to reduce channel contention and packet collision rates. The counter-based method works by inhibiting rebroadcasting if the number of nodes being able to receive the subsequent broadcast transmission is low.

There are many proactive routing protocols proposed for ad-hoc networks. The most extensively cited proactive routing protocol is the Destination-Sequenced Distance-Vector (DSDV) (Perkins & Bhagwat 1994) protocol which was specifically designed for ad-hoc networks, and its operation is detailed in Section 4.4. Furthermore, the proactive Optimised Link State Routing (OLSR) protocol (Clausen & Jacquet 2004), was later developed for ad-hoc networks and was optimised by electing certain nodes to become multipoint relays which aim to reduce routing overhead. The operation of OLSR is detailed in Section 4.5.

4.3.2 Reactive Routing

The alternative to proactive routing is to employ a reactive routing protocol that does not continuously maintain a routing table over time. Instead, it only determines a routing path on-demand as needed by the requesting node. This is referred to as the route discovery process, and this process must be completed before a data packet can be transmitted. As such, the routing table consists of previously learned routing paths discovered through the discovery process, and are retained for some period of time.

The route discovery process can take a considerable amount of time to return a valid routing path. This is especially the case if the source and destination nodes are not located near one another from a topological point of view, i.e., the number of intermediate relay hops is large. The fact that the amount of time can be excessive for a node to discover a valid routing path indicates that a reactive routing protocol is possibly not suitable for very large networks, or networks that carry delay sensitive traffic data. On the contrary, the amount of overhead traffic produced is considerably less than proactive routing protocols, since it does not rely on routing update packets to be periodically transmitted through flooding.

This suggests that the two alternative routing strategies both exhibit problems in terms of scalability – by either producing vast amounts of routing overhead, or incurring lengthy delays. Furthermore, blindly transmitting reactive route request packets across the network without considering already known routing information is

a waste of channel resources. The solution therefore lies in combining the positive aspects of both the proactive and reactive protocols to create a so called hybrid protocol that does not have problems with scalability and routing overhead, and where the route discovery process does not incur long delays.

Numerous reactive routing protocols have also been proposed for ad-hoc networks. In the literature, the most prominent reactive routing protocols are the Dynamic Source Routing (DSR) (Broch, Johnson & Maltz 1998) protocol, and the Ad-hoc On-Demand Distance-Vector (AODV) (Perkins 1997) routing protocol. Both protocols use route request (RREQ) techniques to discover the path to a desired destination node. The former uses a distance-vector scheme similar to DSDV but in the reactive sense, whereas the latter uses a source-routing scheme. An extensive performance comparison of these two reactive routing protocols is given by Perkins, et al. (Perkins et al. 2001).

The DSR protocol is a sound candidate for demonstrating reactive routing protocols (Adibi & Agnew 2008) since the results generalise other reactive routing protocols using similar techniques (Maltz et al. 1999). Also, in research by Ehsan & Uzmi (Ehsan & Uzmi 2004), the authors established that DSR is the more suitable reactive routing protocol for ad-hoc networks, and outperforms AODV in a number of different scenarios with various metrics such as mobility and network size. Overall, DSR was found to generate less routing overhead and experiences lower end-to-end delay, as well as being able to utilise a routing cache. Furthermore, the performance statements made by these authors was verified by using the NS-2 network simulator (NS-2 2005) by comparing the routing overhead for both DSR and AODV using various network topologies and operating scenarios. However, it was noted that the routing cache does expose DSR to routing path stability problems when node mobility is high, leading to the routing cache becoming increasingly stale. An overview of DSR is presented in Section 4.6, and also becomes the motivation for a discussion in relation to source-routing.

4.3.3 Hybrid Routing

Several hybrid routing protocols have been proposed for ad-hoc networks, of which the Zone Routing Protocol (ZRP) (Haas 1997), Zone-based Hierarchical Link State (ZHLS) (Joa-Ng & Lu 1999), and Distributed Dynamic Routing (DDR) (Nikaein, Labiod & Bonnet 2000) are the most common. Of these three, the most recognised hybrid routing protocol is ZRP in which the authors propose a hybrid routing framework suitable for developing hybrid routing protocols.

Two definitions for hybrid routing protocols exist. The first being a combination of static and dynamic routing, and secondly, a combination of proactive and reactive routing. The former is primarily found in infrastructure-based networks in which certain segments of a network are statically routed (where changes in the topology occur infrequently), and other segments are dynamically routed (where changes are more likely to occur). The latter definition of a hybrid routing protocol is predominantly found in wireless networks, whereby a node utilises a proactive approach for its neighbouring nodes within a well-defined "awareness" region, and employs a reactive approach when the destination node falls outside of its awareness region. In this chapter, the emphasis will be placed on the latter definition for a hybrid routing protocol pertaining to wireless networks.

From the discussion in Sections 4.3.1 and 4.3.2, it is found that proactive routing protocols are best suited to small low mobility networks with high bandwidth, whereas reactive routing protocols are best suited to larger high mobility networks with low bandwidth. For this reason, the desired solution would be to combine the positive aspects of both proactive and reactive protocols to create a hybrid routing protocol.

The hybrid strategy is to limit the awareness region of each node to a certain distance in terms of hop-count. Firstly, each node maintains a proactive routing table for a distinct hop-count limit, and any changes in connectivity that occur outside of this region will not affect the routing table. This effectively means that the scope of the proactive protocol is limited to a certain distance. As such, no routing packets will be received by a node outside the awareness region for changes that affect nodes

local to where the change in topology occurred. For that reason, the amount of routing overhead traffic being transmitted on the channel is significantly reduced, at the expense of having a partial or non-global topological view of the network. This characteristic does not generally adversely affect the ad-hoc networks usability from the end-users' point of view. This is because for the majority of time in practice, it is assumed that most nodes will primarily communicate with other nodes close in topological proximity (e.g., communication between colleagues within a building). It should be noted, however, that for the purposes of network performance evaluation, computer simulations using a random waypoint (RWP) model of movement with a uniformly distributed random destination point is used, as discussed in Section 7.2.2. Furthermore, the traffic patterns between nodes within the network are evaluated using both a user datagram protocol (UDP) traffic source and a transmission control protocol (TCP) traffic source, as discussed in further detail in Section 7.2.3.

Secondly, if a node is requesting a routing path to a destination node that falls outside of its awareness region, the reactive approach is taken. As previously discussed, this could potentially lead to large time delays being incurred during the route discovery process. The mechanism by which the reactive process occurs within the hybrid protocol will be described in further detail in Section 4.7, but primarily relies on either a modified flooding technique through broadcasting, or a derivative of flooding. In Chapter 5, a scheme is proposed with which the hybrid routing protocol does not rely on flooding for route discovery, but instead relies on unicasting route request (RREQ) packets to specifically selected nodes, thereby minimising the route discovery delay and routing overhead.

4.4 AN OVERVIEW OF DESTINATION-SEQUENCED DISTANCE-VECTOR (DSDV)

The DSDV protocol is distance-vector based, and makes use of the Bellman-Ford algorithm to determine the shortest routing paths, as discussed in Section 3.3.2. In addition, the counting-to-infinity problem (which gives rise to infinite routing loops) and temporary routing loops has been addressed through the introduction of sequence numbers, to be discussed in Section 4.4.2. The aim of the DSDV protocol is to retain

the uncompounded simplicity of the Bellman-Ford algorithm, yet make it suitable for ad-hoc networks thereby creating the distributed Bellman-Ford algorithm.

4.4.1 Update Packets

In DSDV, the routing table kept at each node contains three fields; namely, the destination node, next-hop node, and the routing metric information for each destination node that is reachable within the network. Since DSDV is proactive, the protocol periodically transmits routing update packets to its neighbouring nodes. To reduce bandwidth usage, which is a major concern of all proactive (and reactive) routing protocols, two types of routing update packets are defined in DSDV. The first type is referred to as a "full dump" packet, which contains the entire routing table known at the node. The second type is known as the "incremental" packet, which contains only the topological changes that occurred since the last update packet was transmitted. It should be noted that not only proactive routing protocols suffer from high bandwidth usage when faced with harsh network conditions. Reactive routing protocols also require the bandwidth usage to be carefully managed especially during frequent topological changes which can cause route cache expiry problems, and these issues are investigated later in Section 4.6.

Since there is no explicit update period defined in DSDV (Perkins & Bhagwat 1994), the frequency of transmitting an incremental packet is left to the decision of each node, but is typically set to 30 seconds as per the RIP protocol (Hedrick 1988). The full dump packet is transmitted every full dump period. This period is determined by each individual node to occur when the incremental packet size reaches the approximate packet size of a full dump packet. In this case, if a large enough fraction of routing table entries had numerous changes occurring, this will be proportionally reflected in the size of the incremental packet. Therefore, it would be of greater benefit to transmit a full dump packet over an incremental packet. In this way, the size of the incremental packets will be smaller for subsequent updates, as overhead grows proportional to the order of $O\left(N^2\right)$, where N is the number of nodes in the network.

In research by Boukerche, et al. (Boukerche, Fabbri & Das 2000), the authors propose a randomised version of DSDV called R-DSDV. They implement a scheme for congestion control through altering the update packet frequency for each node independently using a probabilistic model. In essence, by reducing the frequency of update packet transmissions, an intermediate relay node is able to reduce the amount of traffic forwarded through itself. The author shows that a linear relationship exists between decreasing the frequency of update packet transmissions and congestion experienced at the node. At the same time, the convergence time is found to be short, and is therefore suitable for reducing temporary congestion experienced at each node within the network.

In a work by Lu, et al. (Lu et al. 2003), the authors propose a congestion avoidance scheme for DSDV, called congestion-aware distance-vector (CADV). The proposed scheme attempts to reduce the unfairness in routing path utilisation since DSDV utilises the same route repeatedly leading to load imbalance. This can cause heavy congestion at specific intermediate relay nodes which are frequently used. The CADV scheme works by favouring the routing path with lower queue delay at the MAC level, rather than minimum distance routing as the primary selection criterion. The queue delay information is carried as an additional field in the routing update packet.

In a study by Ahn & Udaya-Shankar (Ahn & Udaya-Shankar 2001), the authors propose an adaptive scheme as an extension to DSDV, called adapting to route-demand and mobility (ARM). With ARM, each individual node maintains a mobility and route-demand metric. By monitoring the number of topological changes, e.g., the mobility metric, the ARM-DSDV protocol can adaptively adjust the frequency of route update transmissions, thus providing fewer update packets when mobility is low. Similarly, the route-demand metric which monitors the usage statistics of nodes, allows the content of the update packets to be dependent on the actual utilisation of the routing information. In this way, the update packet will not contain the unutilised routing paths in every sequential packet. Instead, the routing update packet will only contain updates pertaining to unutilised routing paths every $1/K$ updates, where K is adaptively chosen by each node.

In research by Bansal, et al. (Bansal et al. 2002) and Chaplot (Chaplot 2002), it is observed that DSDV does not perform adequately in high mobility networks, since the routing update packets fail to converge when mobility increases. This is primarily because DSDV has to maintain the routing table for every node within the network, including those with long routing paths. As such, the routing update packets are transmitted at a lower frequency than topological changes occur. Indeed, in a wireless environment the signal-to-noise ratio (SNR), bit-error-rate (BER), and transmission rates can fluctuate rapidly as to cause many connectivity changes, making routing information packets unable to de delivered on time, given the routing update frequency.

In fact, it should be noted that most ad-hoc routing protocols use minimum hop-count as its preferred metric. Having fewer hops between a source and destination node implies a longer physical distance between each consecutive hop. Since longer physical distance paths exhibit higher BER and/or lower transmission rates compared to a shorter physical distance, minimum hop-count routing paths usually demonstrate higher aggregate BER with lower average transmission rates than routing paths with more hops. Conversely, the overall throughput capacity of a multi-hop network is determined by the number of hops in a routing path, given by the order of $O\left(\frac{1}{\sqrt{N}}\right)$, where N is the number of nodes in the routing path of the packet (Li et al. 2001). Therefore, it is apparent that a trade-off exists between link quality and hop-count distance. If a packet is given a longer routing path, or in other words a longer hop-count distance to travel the same physical distance, the throughput capacity is effectively reduced and latency is increased. This holds true when the transmission rate is not increased due to the shorter physical distance travelled, rather the received signal is yielding an improved BER when N is larger.

4.4.2 Sequence Numbers

The approach taken by DSDV to guarantee loop free routing is to assign an additional field of information, called the sequence number, for each destination node within the routing table. Given the three existing fields (destination node

address, next-hop, and the corresponding routing metric previously obtained using route updates), the addition of a sequence number allows the receiving node to determine the integrity of the received update information. The sequence number is originated at the source node, or in other words, each node maintains a sequence number that may only be incremented by itself after each routing update transmission. The sequence numbering scheme in DSDV is defined such that a node must increment to the nearest even number, as an odd number is defined to represent infinity. Therefore, when a node detects a link breakage, it increments the sequence number by one to force an odd number, which accordingly sets the routing metric to infinity.

In principle, the routing update information bearing the highest sequence number will be retained by each node, and the older information will be discarded. However, if the sequence number remains the same, it proceeds to compare the current and new routing metric values. If the new routing metric is better than the existing metric, the information will also be retained. If a significant change is detected within the network, e.g., a new destination node is discovered or has become unreachable; the information is immediately advertised to neighbouring nodes by triggering an incremental routing update packet. Alternatively, if a lower metric routing path is found, this information is advertised to neighbouring nodes only after an "average settling time" has elapsed to account for damping fluctuations (Perkins & Bhagwat 1994).

The average settling time is defined as the amount of time between the arrival of the first and the best routing path carrying the same sequence number. By preventing nodes from broadcasting the lower routing metric for twice the average settling time since the first arrival, it helps alleviate routing fluctuations. This process is illustrated in the following example. Let node **A** transmit an update packet with sequence number 500 to both nodes **B** and **E**, as shown in Figure 4.2. Assume there are X hops between nodes **B** and **C** that are highly congested with traffic, and $X+1$ hops between nodes **E** and **F** that are relatively idle. As a result, node **D** will receive the update packet with sequence number 500 from node **F** first, and in response must propagate its own update packet to its neighbours. However, a short time later (due to the congestion) node **D** will receive the same update packet with sequence number

500 with fewer hops from node **C**. As a result, node **D** will be required to rebroadcast its update packet again announcing the lower metric. This scenario can be avoided if node **D** has to wait twice the average settling time before broadcasting the lower metric, giving a window of time to receive all update packets.

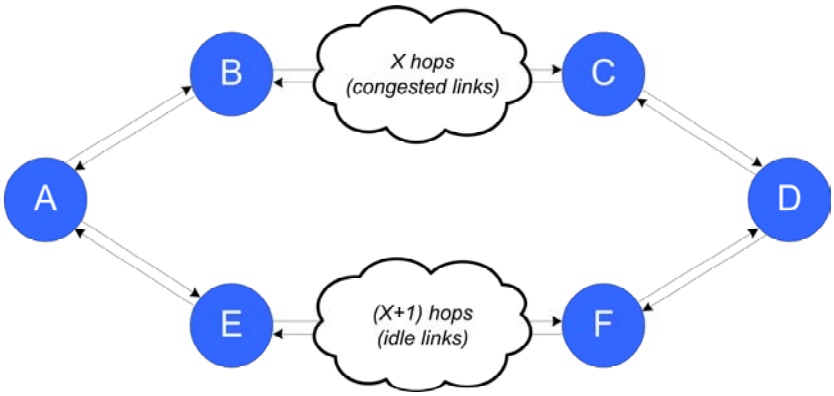


Figure 4.2 – DSDV scenario susceptible to routing metric fluctuations.

Furthermore, in a study by Zhao, et al. (Zhao et al. 2005), it is suggested that problems can arise if several intermediate relay nodes do not receive the best routing metric to begin with. Consequently, the delay experienced in determining the shortest path is the cumulative summation of twice the average settling time for all intermediate relay nodes.

In DSDV, the receipt of a new sequence number is not regarded as a significant change, and therefore will not trigger an incremental routing update packet. A modification called DSDV sequence number (DSDV-SQ) is proposed by Broch, et al. (Broch et al. 1998). It exploits the fact that a node which detects a link breakage will immediately trigger an incremental routing update packet to be sent to its neighbouring nodes. However, these nodes will not propagate this information further, according to standard DSDV, given that only the sequence number would have changed. In DSDV-SQ, however, the arrival of a new sequence number at the neighbouring nodes is regarded as a significant change, and will cause a proliferation of incremental update packets to be transmitted. The end result is that link breakages are detected quicker and therefore not utilised in the route selection algorithm. In turn, the opportunity arises with which a node is able to circumvent network damage by utilising alternative routing paths.

Moreover, the property of sequence numbers allows each node the capability to remove stale nodes as they arise. A stale node occurs when a node stops receiving an incremented sequence number from a particular destination node. In this way, it can be determined that a particular destination node is no longer reachable from the current node, and hence the current node will remove the destination node entry from its routing table.

4.5 AN OVERVIEW OF OPTIMISED LINK STATE ROUTING (OLSR)

The OLSR protocol is a proactive link-state based protocol that exchanges topological connectivity information between neighbouring nodes periodically. It uses a technique called hop-by-hop routing such that packets do not carry a source-route within their packet headers. Instead, each intermediate relay node uses its local routing table to guide and forward a packet towards the destination node. OLSR makes use of two techniques to acquire topological information in order to build a routing table at each node, and these are discussed hereafter.

4.5.1 Multipoint Relays

Multipoint relay (MPR) nodes are responsible for forwarding and disseminating routing traffic within an ad-hoc network. The MPR nodes are selected from the set of neighbouring nodes located 1 hop away from each node within the network. There are two rules which govern the selection of an MPR node, firstly, any node located 2 hops away must be covered by at least one MPR node, and secondly, the set of MPR nodes selected must be minimised in order to reduce the routing traffic overhead. Furthermore, a packet is not forwarded if the same packet has been recently received from a neighbouring node through the use of a sequence number detection algorithm.

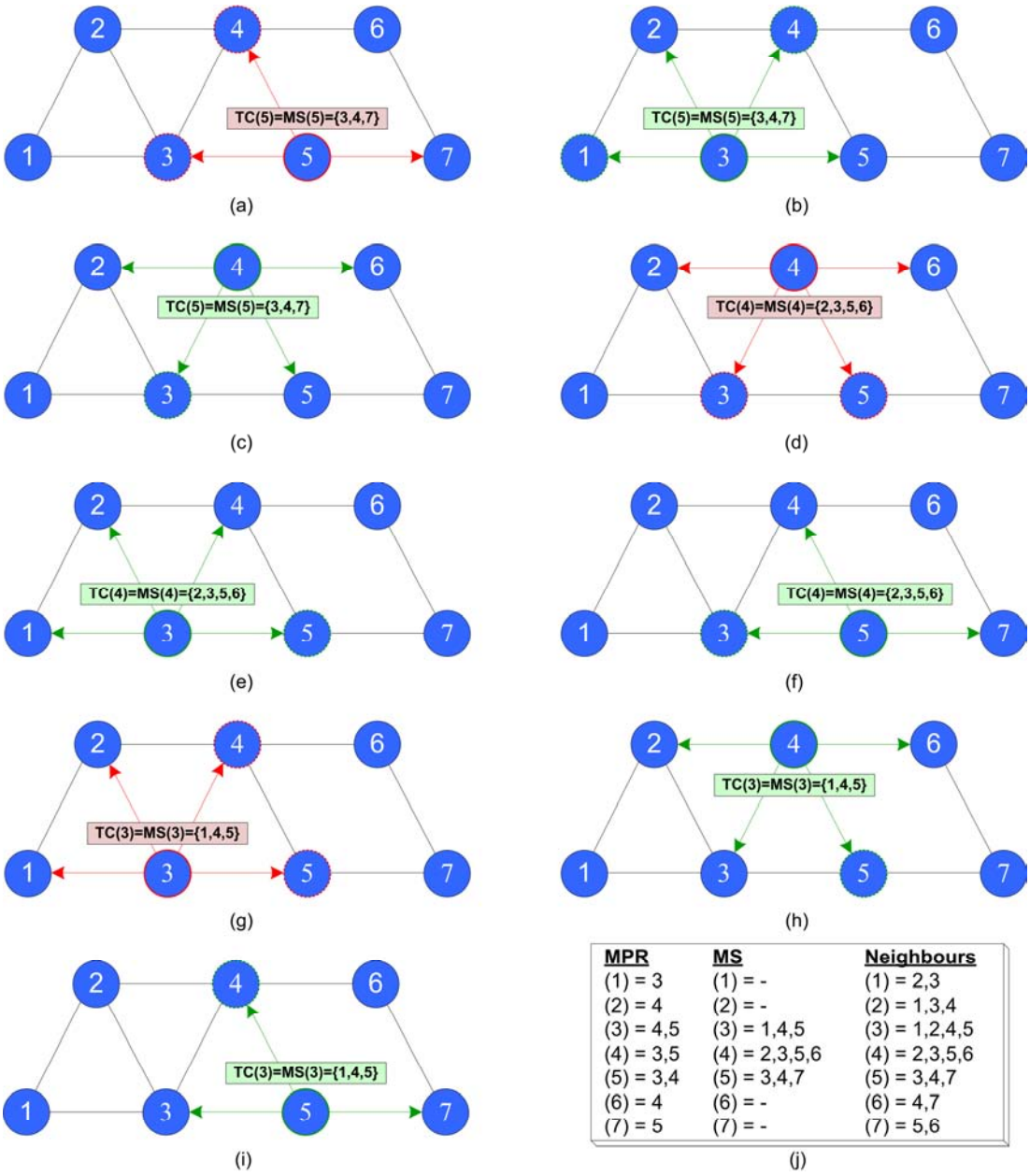


Figure 4.3 – An example of OLSR topology discovery procedure using multipoint relay nodes to forward topology control packets. Parts (a–c) show the step-by-step TC packet transmission for node 5 (shown in red) and the subsequent forwarding by nodes 3 and 4 (shown in green), parts (d–f) and (g–i) show the TC packet transmissions for nodes 3 and 4, respectively, and (j) shows the MPR, MS, and neighbouring nodes list for each node within the ad-hoc network.

Initially, the neighbouring nodes are discovered through the use of periodic "hello" packets being broadcast. In the topology shown in Figure 4.3, it is observed that

node 5 has three neighbouring nodes that are located 1 hop away, namely nodes {3,4,7}. Of these neighbouring nodes, node 3 has further connectivity with node 1, nodes {3,4} have further connectivity with node 2, and node 4 has further connectivity with node 6. As a result, the minimum MPR set that node 5 can select is MPR(5)={3,4} in order to retain connectivity with nodes {1,2,6} located 2 hops away. The method by which node 5 discovers this set is depicted in Figure 4.3(a–i) and is discussed below in Section 4.5.2.

The introduction of MPR nodes aims at reducing the routing traffic overhead by minimising the negative effects of flooding. Accordingly, only the MPR nodes participate in the transmission of link-state changes with the nodes that selected them within the network. In other words, an MPR node periodically transmits routing update packets that indicate which neighbouring nodes it can reach to the nodes that selected them as MPR, termed the MPR selector (MS) set.

In the case of Figure 4.3(a), the MPR selector set of node 3 must include node 5, i.e., MS(3)={5,…} and similarly for node 4 that MS(4)={5,…}. Furthermore, a novel technique used in OLSR is the use of a *Willingness* field, such that a node can choose to always or never be selected as an MPR node, as opposed to being selected by a neighbouring node.

4.5.2 Topology Discovery

In OLSR, the discovery of routing paths is carried out through the dissemination of link reachability information between nodes and their selected set of MPRs using "hello" packets at 2 second intervals. In the example shown in Figure 4.3, node 5 will broadcast a hello packet to advertise that it is aware of nodes {3,4,7}, and this packet is not further broadcast as it has a TTL of 1. In turn, the other nodes will advertise their neighbouring node knowledge according to the table shown in Figure 4.3(j). From this sequence of transmissions, node 5 can determine the optimal (or near optimal) MPR set by calculating that the minimum set is equal to {3,4} to maintain reachability with nodes {1,2,3,4,5,6,7}, and the interested reader is referred

to the OLSR protocol specification document (Clausen & Jacquet 2004) for the MPR computation algorithm.

An MPR node must transmit topology control (TC) information at 5 second intervals advertising its MS set and an increasing sequence number to prevent stale information. The TC packet is received and processed by all the neighbouring nodes located within 1 hop of the transmitter. Only those selected MPR nodes will forward the TC packet to disseminate topological information within the network, or in other words, a node only forwards the TC packet if the transmitter is an element of its MS set. This process is shown in Figure 4.3(a), where node 5 broadcasts its TC(5)={3,4,7} packet which is received by nodes {3,4,7} and is subsequently forwarded by nodes 3 and 4 (as MPR(5)={3,4}) to the remainder of the network, as shown in Figure 4.3(b–c). This process is continued for node 3 with its TC(3) packet, as shown in Figure 4.3(d–f), and for node 4 with its TC(4) packet, as shown in Figure 4.3(g–i).

It is noted that nodes {1,2,6,7} do not transmit a TC packet because their MS set is empty, and thus don't advertise any topological information apart from their hello packet to their immediate neighbours. From this example, it is apparent that all the nodes are now able to reach any other node within the network by using the advertised link-state information encapsulated within the TC packets and creating a corresponding routine table.

In Figure 4.4, the non-advertised links are only visible to their respective nodes, such that the dashed link between nodes 1–2, 2–3, and 6–7 are only visible to themselves. From the TC packet transmissions shown in the previous example, all the nodes within the network can generate a routing table using the link-state information received. As an example, the routing table for node 7 is depicted in Figure 4.4 indicating the next uplink node used to reach a destination node, including the hop-count metric associated with that routing path.

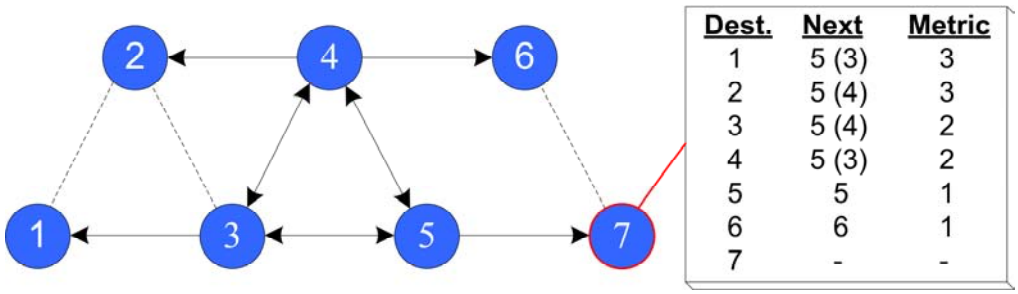


Figure 4.4 – Example OLSR topology showing the advertised links transmitted by the MPR nodes during the topology discovery procedure (shown by solid arrows), and indicating the links that are not advertised (shown by dashed lines). The corresponding routing table for node 7 is shown.

Through the use of the MPR and MS set information, the bandwidth used to propagate routing control information is reduced in two ways. This is achieved through reducing the distance the hello packets are broadcast using a TTL of 1, and secondly, by reducing the number of transmissions required to disseminate TC packets (when compared to flooding) by targeting specifically selected MPR nodes with the use of sequence numbers to prevent redundant retransmissions.

4.6 AN OVERVIEW OF DYNAMIC SOURCE ROUTING (DSR)

The DSR protocol is path-vector based, which is also known as source-routing. With DSR, the routing path is entirely specified from the source node to the destination node. However, the routing paths are non-deterministic in the sense that a discovered routing path will not necessarily be the same path every time. As a consequence of source-routing, DSR is implicitly capable of providing loop-free routing due to its knowledge of all the nodes within the routing path through which a packet must traverse.

For its route cache, the DSR protocol uses a simple “path cache” where it can store a known routing path to a given destination. Alternatively, the path cache can also be represented by a “link cache” in DSR, so that Dijkstra’s algorithm may be used to determine the shortest routing path. However, this is not computationally efficient compared with the path cache method (Maltz et al. 1999). An analysis of different

route cache and link cache algorithms involving different mobility models has been presented by Hu & Johnson (Hu & Johnson 2000). The analysis was further improved by Wu, et al. (Wu, Hou & Hou 2003) by taking into consideration the signal strength and route time stamps in the caching algorithms. This additional information is used in order to purge invalid or old routing paths from the routing cache, however, at the expense of a large increase in overhead which is the focus of continuing research.

Since the DSR protocol is purely reactive, it does not require any periodic routing update packets to be transmitted. Instead, the DSR protocol is comprised of two main parts to determine the network topology information; namely, route discovery, and route maintenance.

4.6.1 Route Discovery

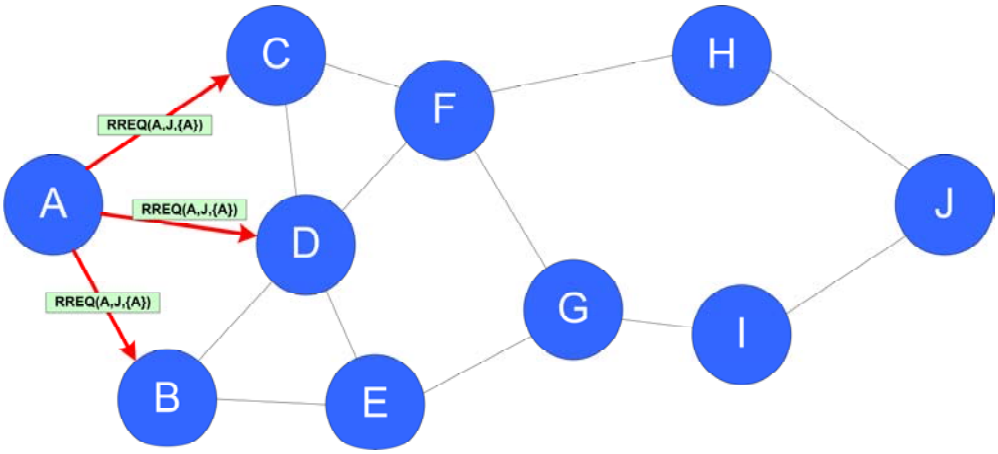
Route discovery is the process of determining a routing path between a source node and a destination node when such a path is not already known. During route discovery, a route request (RREQ) packet is first transmitted. This packet contains four fields; the source address, destination address, route request identification number, and a list of intermediate relay nodes through which the RREQ has traversed. The RREQ packet is disseminated to the neighbouring nodes by means of flooding, and is influenced by a propagation distance control scheme and a RREQ identification number scheme.

To control the distance of propagation in terms of hop-count, DSR uses an "expanding ring search" (Lee, Belding-Royer & Perkins 2003) technique similar to AODV. With this technique, the node that initiates the transmission of a RREQ packet is responsible for setting the time-to-live (TTL) of a packet with an initial starting value. This value is then gradually incremented when no route is found after a certain amount of time, called the RREQ timeout. However, the use of an expanding ring search could increase the average latency of route discovery, especially if a destination node is located far away. In such a case, multiple RREQ packets would have to be transmitted resulting in multiple RREQ timeouts. When

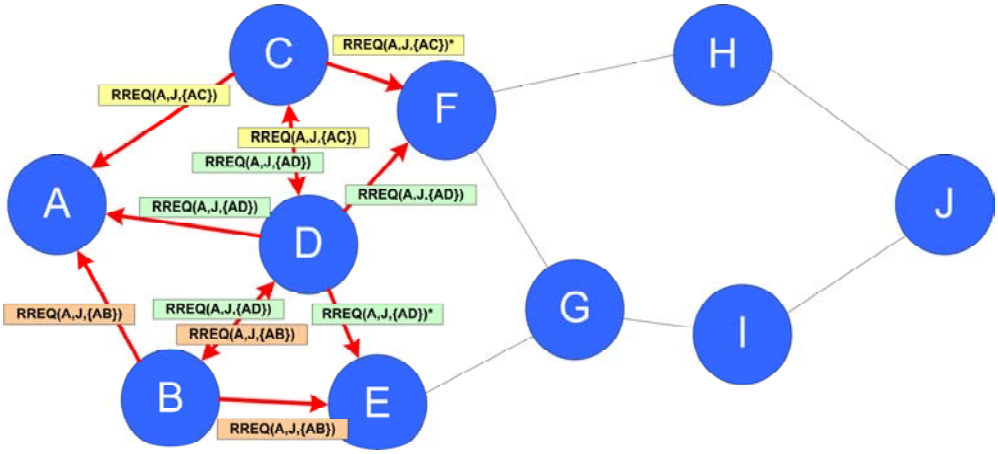
different TTL values are used for the expanding ring search, it is found that the routing overhead is only marginally reduced but the route discovery latency is significantly increased (Cheng & Heinzelman 2003; Koutsonikolas et al. 2005). On the other hand, the best performance is achieved using a two-tier expanding ring search with a TTL value of $N/2$ in the first round, where N is the maximum hop-count routing path of a given network. This is followed by a second round of the expanding ring search with a TTL value of N.

The use of the RREQ identification number allows an intermediate relay node to determine whether or not it has previously come across the same RREQ packet. This event can occur when another intermediate relay node forwards a RREQ packet through flooding, hence, the packet can propagate backwards towards a previously encountered node. In this case, if the same RREQ packet has previously been received by the intermediate relay node, it will discard it to prevent repetitive propagation of the RREQ packet. It should be noted that in DSR if two or more RREQ packets arrive at a given node following different paths, only the first arriving packet will be accepted while all the latter packets will be discarded regardless of whether or not their accumulated paths are shorter.

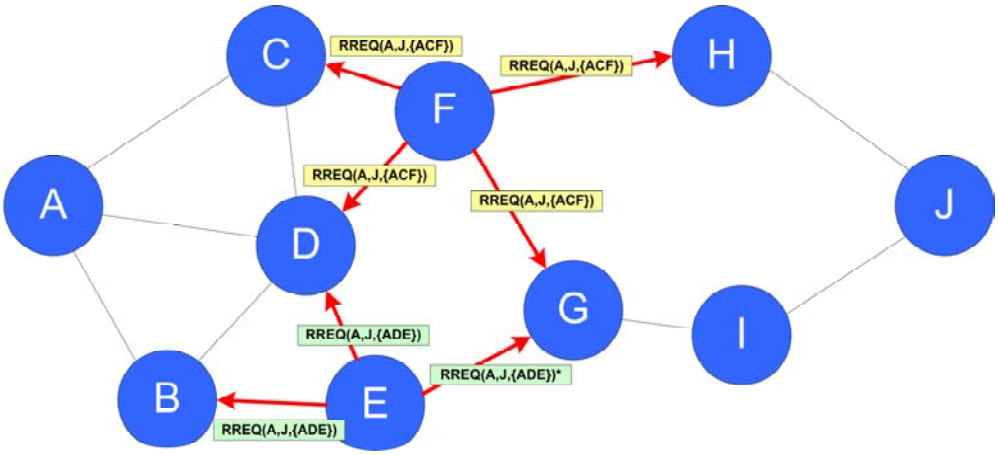
The RREQ process illustrating the dissemination of a RREQ packet by means of flooding to neighbouring nodes is shown in Figure 4.5(a–d). The process is initiated by node **A** transmitting a RREQ packet to its surrounding nodes, namely, nodes **B**, **C**, and **D**. The arrival of the RREQ packet at these nodes triggers a subsequent transmission from these nodes to their surrounding nodes, namely, nodes **E** and **F**. It should be noted that the surrounding nodes that have already received the RREQ packet previously will discard the packet to prevent repetitive propagation of the RREQ packet. The RREQ packet is transmitted again and received by nodes **G** and **H**. It is at this point when node **H** transmits its RREQ packet to its surrounding nodes (namely node **J**), causing a route reply (RREP) packet to be generated, which is illustrated in Figure 4.6(a–b).


C
H
F
RREQ(A,J,{A})
A
RREQ(A,J,{A})
J
D
RREQ(A,J,{A})
G
I
B
E

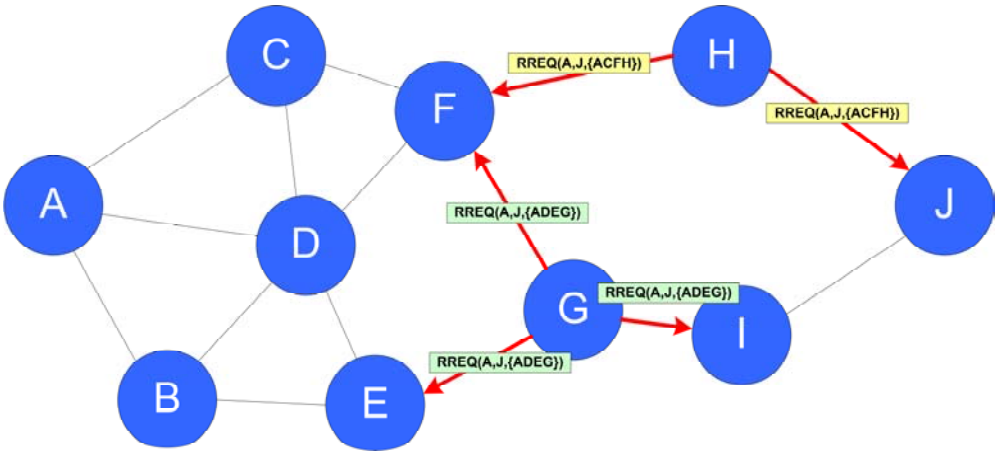
(a)


C
RREQ(A,J,{AC})*
H
RREQ(A,J,{AC})
F
RREQ(A,J,{AC})
RREQ(A,J,{AD})
A
RREQ(A,J,{AD})
RREQ(A,J,{AD})
J
D
RREQ(A,J,{AB})
RREQ(A,J,{AD})
RREQ(A,J,{AD})*
G
RREQ(A,J,{AB})
I
B
E
RREQ(A,J,{AB})

(b)


C
RREQ(A,J,{ACF})
H
RREQ(A,J,{ACF})
F
RREQ(A,J,{ACF})
A
RREQ(A,J,{ACF})
J
D
G
RREQ(A,J,{ADE})
I
RREQ(A,J,{ADE})*
B
E
RREQ(A,J,{ADE})

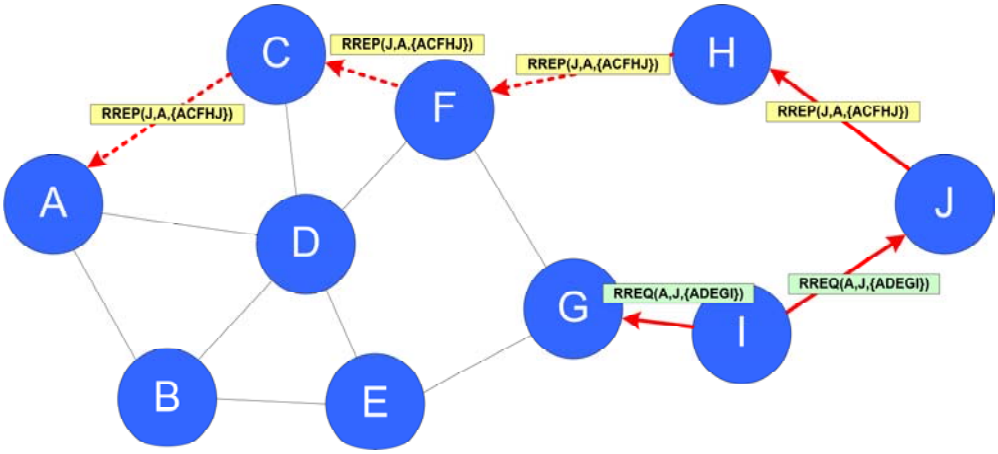
(c)



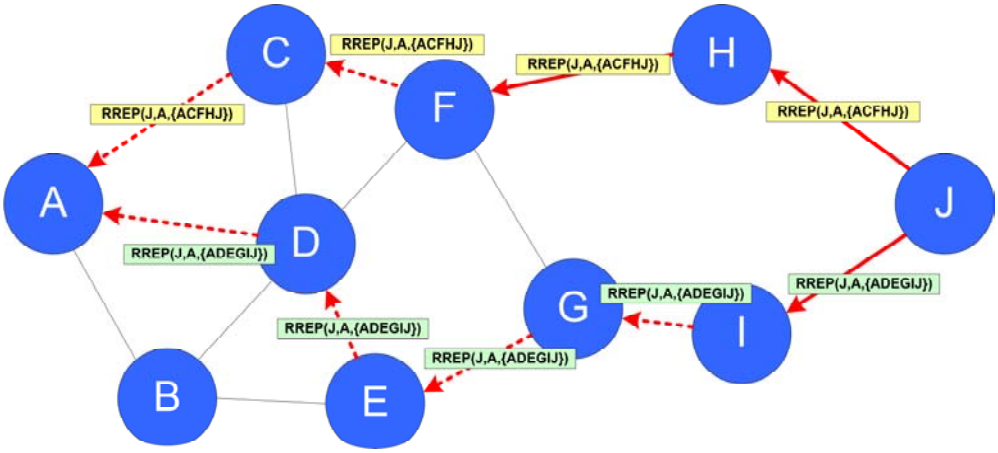
(d)

Figure 4.5 – Example DSR route request, (a) shows the initial transmission from node **A** to its surrounding nodes (shown in green), (b) showing the subsequent transmissions from nodes **B**, **C**, and **D** (shown in pink, green, and yellow), (c) showing the subsequent transmissions from nodes **E** and **F** (shown in green and yellow), and (d) showing the final route request transmission by node **H** before a route reply is generated by node **J** (shown in yellow, and in the next figure). The transmission process by node **G** is illustrated in the next figure.

During the RREQ process, whenever an intermediate relay node is unaware of the destination node, it will append its address into the list of intermediate relay nodes in the RREQ packet, and then propagate the packet through flooding. Alternatively, if the intermediate relay node is aware of the destination node, it will then transmit a route reply (RREP) packet back towards the initiating source node. This can be performed in two ways, the first being route reversal which assumes bidirectional links. This approach involves reversing the traversed route in the list of intermediate relay nodes back to the source. The second approach is to initiate a route discovery process directed back towards the source node.



(a)



(b)

Figure 4.6 – Example DSR route reply, (a) shows the RREP packet transmitted form node **J** back to node **A** with a routing path of 5 hops whilst the RREQ is still progressing in another part of the network, and (b) showing the additional non-standard DSR route reply with a routing path of 6 hops (non-optimum) that occurred shortly afterwards.

When two or more RREQ packets arrive at the destination node after following different paths, the latter packets will not give rise to RREP regardless of whether or not their accumulated paths are shorter than that of the first arriving RREQ packet. This particular characteristic of DSR could result in the source-route obtained by the RREQ process to be non-optimum, due to a non-minimum hop-count routing path.

An example of such an event is when the RREP, as shown in Figure 4.6(b), occurs before the RREP shown in Figure 4.6(a). As observed in the RREQ example shown in Figure 4.5, the first RREQ packet to arrive at node **J** is from node **H** containing the accumulated path "**A-C-F-H**". This will trigger a RREP packet to be transmitted by node **J** back to node **A** with the complete routing path specified as "**A-C-F-H-J**", as shown in Figure 4.6(a). In addition, second RREQ packet will arrive at node **J** from node **I** containing the accumulated path "**A-D-E-G-I**", as shown in Figure 4.6(b). If this RREQ packet was received first by node **J**, it will instead trigger a RREP packet with a routing path of "**A-D-E-G-I-J**" to be transmitted back to node **A**, which is one hop-count longer and thus non-optimum.

A beneficial side-effect of RREP packet transmission is that intermediate relay nodes forwarding a RREP packet have an opportunity to extract the source-route information for use in their own route cache. Furthermore, nodes neighbouring the intermediate relay nodes that are eavesdropping on the channel, called promiscuous listening mode, also have an opportunity to extract the source-route information. As a result, a single route discovery process can result in many nodes learning the topological information of the network, hence reducing the likelihood for further RREQ packets to be transmitted. However, this can lead to unfairness in two respects as found by Miranda & Rodrigues (Miranda & Rodrigues 2005). Firstly, an intermediate relay node that is aware of the destination node will transmit a RREP using an existing source-route from its own cache, even though other possible alternative routes might exist that are as yet undiscovered. Secondly, when an intermediate relay node promiscuously extracts the source-route from the RREP packet, it will clearly store the same routing paths in its route cache. This can result in load imbalance in terms of DSR converging to the utilisation of the same few discovered routes, even though possible alternative routing paths might exist but were never discovered.

An enhancement to DSR, called multiple source-routing (MSR), is proposed by Wang, et al. (Wang et al. 2001). It allows the source node to adaptively distribute its traffic load among all the alternative routing paths available. With the MSR protocol, multiple routes can be accumulated and stored in the route cache of a node. In the event of a link failure, other known alternative routing paths could be adopted, thus

avoiding an additional RREQ to be transmitted. The redundancy available within the alternative routing paths is highly beneficial to the overall success of the MSR protocol. However, if alternative routing paths share some common intermediate nodes, this can lead to the breakdown of all routing paths if any of the common nodes fail. It is therefore desirable for alternative routing paths to be selected from among routes which do not share common intermediate relay nodes, except for the source and destination nodes. This will minimise the probability of a routing path breaking down. In other words, the alternative routing paths must be mutually exclusive or disjoint paths. It is shown in a study by Leung, et al. (Leung et al. 2001), that the use of disjoint paths is likely to enhance the reliability of end-to-end connectivity. As a result, it introduces a notion of a soft Quality-of-Service (QoS) guarantee to routing. This can be achieved by allowing an application to specify the end-to-end reliability requirements in order to control the probability of routing failure, and is implemented as an extension to DSR.

An alternative to the MSR protocol is the backup source-routing (BSR) protocol proposed by Guo & Yang (Guo & Yang 2002) and Guo, et al. (Guo, Yang & Shu 2005), in which two different disjoint source-routes are included in each RREP packet. In this way, should the primary routing path fail, the alternative path is immediately available. However, there are four main shortcomings associated with this scheme. Firstly, the specified alternative path could be as stale or staler than the primary path, whereby a stale path is defined in Section 4.4.2. Secondly, every packet is burdened with carrying two source-routes thus increasing the packet overhead. Thirdly, if a multiple source-routing scheme is used to adaptively distribute the traffic load, then the primary and alternative paths specified in the header will toggle back and forth between the only two routes the node is aware of. Therefore, if one of the disjoint routing paths should fail, it would fall-back to conventional DSR but with the burden of additional overhead present in the RREP packet. Finally, a major constraint with this scheme is that it succeeds only if the failure occurs at a node in common between the primary and alternative path, thereby invalidating the disjoint path condition. In other words, since the primary and alternative paths are disjointed, there is no guarantee that the intermediate relay node at which the failure occurred can reach the alternative intermediate relay node without further route discovery. As such, this scheme relies on knowing where the

link failure is likely to occur a priori, and then supplying a backup route for that specific intermediate relay node.

A different approach is described by Wu (Wu 2002), which proposes an extension to DSR by maintaining two disjoint paths in the route cache for each intermediate relay node. This may be achieved by allowing DSR to return multiple RREP packets from a single route discovery process, as previously shown in Figure 4.6(b). Therefore, it becomes possible for an intermediate relay node to utilise an alternative backup routing path by examining its route cache, should the primary routing path fail.

4.6.2 Route Maintenance

Route maintenance provides the ability for repairing a given source-route during its use, or when there are topological changes occurring. Whenever an intermediate relay node is presented with a packet for transmission, be it either a data packet or a routing control packet, DSR checks whether or not the next-hop node is reachable, as specified by the source-route. The verification of the next-hop node being able to receive the packet is an indication that the source-route is still valid. However, if the next-hop is unreachable, a route error (RERR) packet is generated and transmitted back to the source node that supplied the erroneous source-route. The RERR packet contains information on where the broken link has occurred, and this will prompt the source node to reinitiate a RREQ process. In the case of a RREQ packet causing an unreachable next-hop scenario, it will not trigger the above verification process as it is of an exploratory nature.

Now, after the intermediate relay node has determined that the next-hop is unreachable, it will temporarily buffer the packet with the invalid source-route. Since DSR allows for multiple routes to the same destination to be stored in route cache of an intermediate relay node, it becomes possible for the node to repair the source-route thus avoiding the broken link, instead of dropping the packet. With DSR, this correction process is termed packet salvaging, and is performed only after a RERR packet is generated. A direct benefit of packet salvaging is to avoid requiring the intermediate relay node to initiate another RREQ packet for

determining a routing path to the destination. Thus, route discovery using RREQ can be delayed until all alternative routes in the route cache are exhausted through failure. Meanwhile, the number of attempts of salvaging a packet is limited in order to refrain from endlessly salvaging a packet. However, since DSR does not implement path expiration, the alternative path chosen has the possibility of being invalid pre-emptively, or in other words, the alternative path may be stale. It is observed that for networks with high mobility, where link changes occur frequently, it is possible that use of the packet salvaging scheme may exacerbate the problem of utilising stale alternative paths (Valera, Seah & Rao 2005).

The scenario of a packet being transmitted with a source-route containing unreachable intermediate nodes can become worse when the source-route is sufficiently long. If the number of intermediate hops is increased, there is a higher probability of source-route failure. Furthermore, high mobility rates will substantially increase the number of link failure occurrences, as the discovered source route will not remain valid as nodes are being rapidly disassociated, thus proliferating the problem. These series of events will cause numerous RERR packets to be generated, and subsequently cause high route discovery latency due to the flooding of broadcasted RREQ packets. Furthermore, a stale route is removed from the route cache only when a RERR packet explicitly instructs it to do so, as routing paths remain valid indefinitely due to the reactive nature of the DSR protocol. It is observed that DSR could benefit from removing stale routes automatically after some expiry period to prevent cache pollution (Perkins et al. 2001). Another proposed solution is to increase the scope of the RERR packet, i.e., propagate the RERR packet through flooding to obtain an increased notification area, as opposed to unicasting the RERR back to the source node (Marina & Das 2001).

Overall, the DSR routing protocol does not change or optimise the routing path by minimising the hop-count once it has obtained one through the route discovery process. Over time, a routing path may experience degradation in performance due to many factors, e.g., an increase in network congestion which can lead to reduced throughput. In particular, if a shorter routing path becomes available due to the mobility of intermediate relay nodes, DSR will not become aware of this information. In fact, DSR will persist with the use of the first discovered routing path until a

RERR occurs. The use of promiscuous listening to obtain fresh routing information for optimising existing source-routes on-the-fly, if a shorter path can be found, has been proposed by Wu, et al. (Wu et al. 2000). However, such a technique could prove to be unfavourable as it could cause route pollution in terms of intermediate relay nodes re-learning stale information (Hu & Johnson 2002). In other words, it becomes increasingly difficult to remove a genuine stale node from the route cache. Moreover, an enhancement to DSR using a local route discovery process to repair broken routing paths (before a RERR packet is generated and transmitted back to the originating source node) has been proposed by Castaneda & Das (Castaneda & Das 1999) and extended by Castaneda, et al. (Castaneda, Das & Marina 2002). This is based on the fact that an intermediate relay node issuing a RREQ to find a nearby node would receive a RREP faster than if the originating source node had to issue to RREQ to determine a completely new source-route. This idea has been further reinforced by Wu, et al. (Wu et al. 2000) who concludes that a broken node that lies nearby the intermediate relay node could not have moved far, in terms of spatial distance.

4.7 AN OVERVIEW OF ZONE ROUTING PROTOCOL (ZRP)

The ZRP protocol is a hybrid protocol, which includes characteristics of both proactive and reactive based routing protocols. With ZRP, the awareness region is referred to as a zone, which is formed by the set of nodes located within a radius of R hops. The radius is centred at each node from the nodes own perspective, producing a network of N overlapping zones, where N is the number of nodes within the network. Within a given zone, ZRP utilises a proactive routing protocol which is termed the intra-zone routing protocol (IARP). However, when a routing path to a destination node falls outside of this zone, a reactive routing protocol is utilised, and this is termed the inter-zone routing protocol (IERP). Notably, ZRP reduces to purely proactive if $R \to \infty$, and it reduces to purely reactive if $R = 1$. Thus, the actual value of R used determines the reactivity behaviour of the protocol.

The ZRP protocol is described as a framework, such that a specific implementation of the IARP and IERP is not explicitly specified. Instead, research by Haas &

Pearlman (Haas & Pearlman 1998b) recommends the use of existing proactive and reactive routing protocols to perform the functions of the IARP and IERP, respectively. The use of DSDV and DSR has been suggested in Wu, et al. (Wu et al. 2000) as these protocols exhibit characteristics suitable for wireless ad hoc networks, as discussed in Section 4.4 and Section 4.6. The ZRP protocol involves two procedures, namely, route acquisition/discovery, and selective bordercasting which are discussed in the follow sections.

4.7.1 Route Acquisition/Discovery

Route discovery within the zone uses the IARP protocol, which is suggested to be a modified version of the DSDV protocol made suitable for use in zone routing as indicated by Wu, et al. (Wu et al. 2000). The modification necessary to make DSDV zone aware is to inhibit the propagation of routing update packets past a certain hop-count radius R. This measure effectively limits the distance a routing update packet traverses the network through flooding. This strategy can be implemented through means of setting the time-to-live (TTL) field of the routing update packet to $R-1$. As a result, the packet will travel at most R hops before being discarded when the TTL equals zero, as the TTL is decremented at each intermediate relay node. Therefore, using a modified DSDV protocol as the IARP protocol, it allows a source node to proactively maintain a routing table that contains reachability information for each node within that zone.

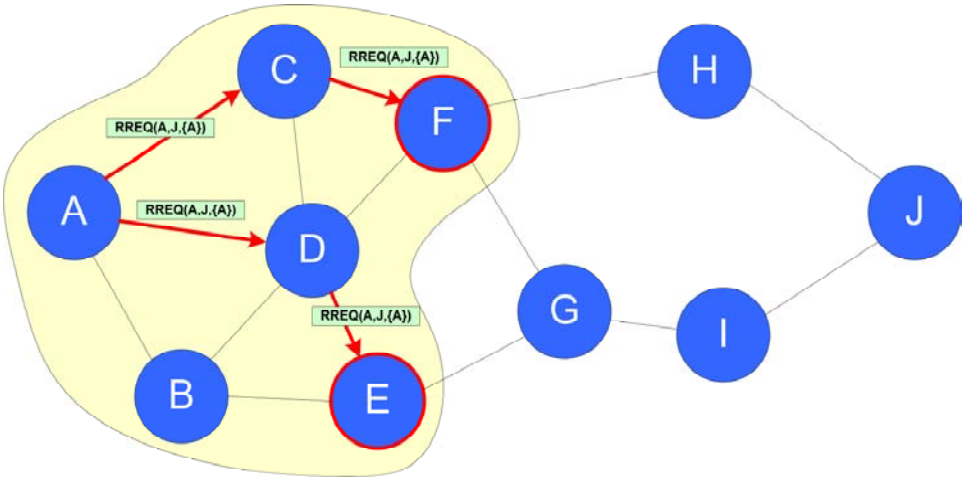
The routing overhead per node is reduced as a consequence of limiting the distance a routing update packet may travel. Since routing overhead for the proactive DSDV protocol grows proportional to $O\left(N^2\right)$ (Lopez, Barcelo & Garcia-Vidal 2005), where N is the number of nodes in the network, it reduces to $O\left(R^2\right)$ since the maximum topological scope of the network for each node is limited to $N_{zone} = R$. As a consequence of this, the full dump and incremental update packets that DSDV periodically transmits, as discussed in Section 4.4.1, also reduce in packet size and frequency as the number of topological changes within the zone are reduced since $R < N$.

For route discovery outside of the zone, the IERP protocol is suggested to be a modified version of DSR as indicated by Wu, et al. (Wu et al. 2000). Since DSR is purely reactive, it will transmit a RREQ packet to discover the routing path to the destination node if the routing path is not already available in the routing table. In order to reduce overhead, ZRP exploits the contents of the routing table provided by the IARP to improve the efficiency of the RREQ and RREP mechanisms in the IERP through a process called "bordercasting" (Haas & Pearlman 1998b).

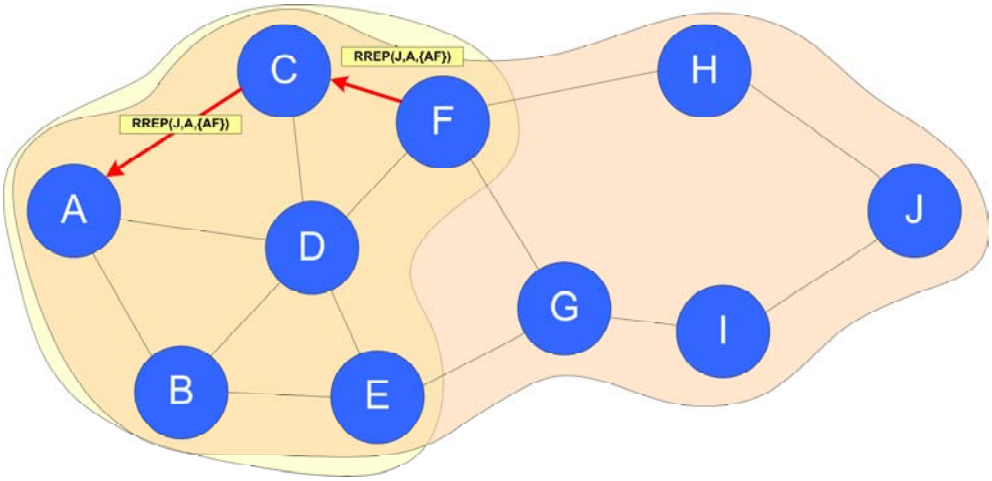
Bordercasting is a packet delivery service provided by the Bordercast Resolution Protocol (BRP). Unlike RREQ flooding in DSR, BRP proceeds to guide RREQ packets away from the source node towards the nodes that define the border of the zone, also known as the peripheral nodes. Even though the RREQ packet traverses through the intermediate relay nodes, the routing path only specifies the peripheral relay nodes in the source-route, and not the intermediate relay nodes themselves. Accordingly, ZRP does not explicitly specify the entire routing path (i.e., source-routing) between the source and the destination node within a RREQ or RREP packet as in DSR. According to Haas & Pearlman (Haas & Pearlman 1998b), bordercasting provides a more efficient method to discover the routing path to a destination node compared to simple flooding through broadcasting.

In terms of protocol operation, ZRP first checks whether or not a routing path to the destination node is available in the routing table provided by IARP for the current zone. If a routing path is not found, ZRP will bordercast the RREQ packet to all the peripheral nodes, since the peripheral nodes act as uplink nodes for the adjacent zones. It should be noted that only the peripheral node addresses are appended in the route header of a RREQ packet, and not the entire path between the source node and each peripheral node (Wu et al. 2000). Each RREQ packet is then bordercast through a series of unicast transmissions to all the peripheral nodes (Haas & Pearlman 1998b). Each peripheral node performs the same bordercasting process until the destination node is found within the IARP routing table of the other peripheral nodes. Once the destination node is found, the peripheral node responds by transmitting a RREP packet back towards the initiating source node. As per DSR route discovery, as discussed in Section 4.6.1, this can be performed using two methods. The first being route reversal, and the second approach is by initiating a

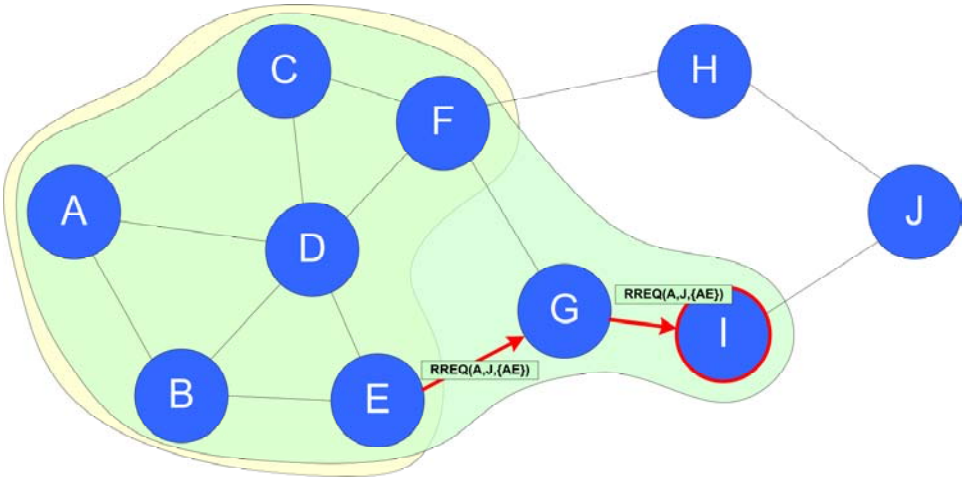
route discovery process back to the source node. The bordercasting process is illustrated in Figure 4.7(a–d), where the peripheral nodes are indicated by a red circle.



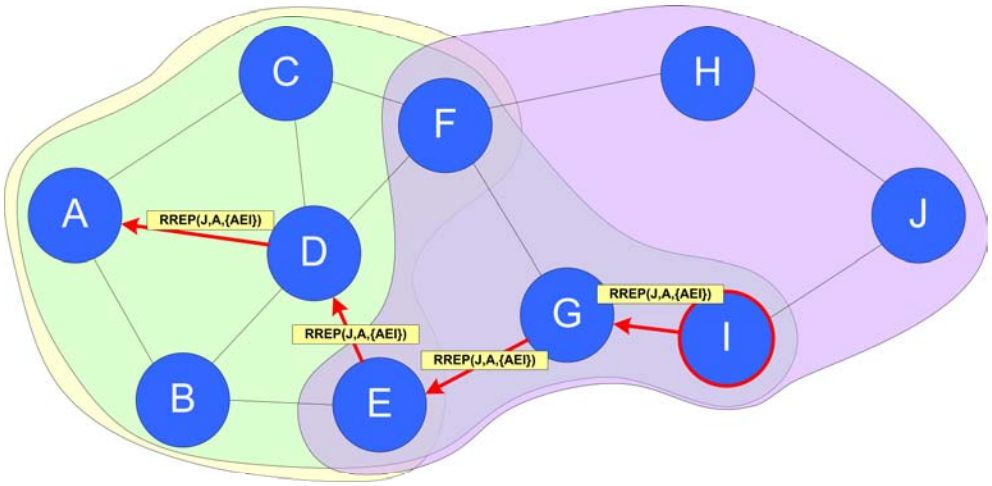
(a)



(b)



(c)



(d)

Figure 4.7 – Example ZRP bordercasting procedure performing a route request, (a) shows the initial transmission from node **A** to its peripheral nodes **E** and **F**, (b) showing the subsequent route reply generated by node **F** back to node **A**, (c) showing the subsequent transmission of the route request packet from node **E** to its peripheral node **I**, and (d) showing the final route reply packet generated by node **I** back to node **A**. The return path is via the peripheral nodes accumulated within the RREQ packet as they were traversed during the route discovery process.

In Figure 4.7(a), the process is initiated by node **A** transmitting a RREQ packet to its peripheral nodes, namely, nodes **E** and **F** to find the destination node **J**. The two individual packets take different routing paths to reach these peripheral nodes, but this path is not stored within the RREQ packet. Instead, only the peripheral node numbers are accumulated within the RREQ packet as the nodes can already determine the best routing path towards these nodes. The arrival of the RREQ packet at the peripheral nodes triggers a subsequent RREQ packet transmission at node **E** as it's unaware of the destination node **J**, whereas node **F** is aware of the destination node **J** and transmits a RREP packet back to node **A**, as shown in Figure 4.7(b). The RREQ packet transmitted by node **E** is received by its peripheral node **I**, which is aware of the destination node **J**, as shown in Figure 4.7(c), and proceeds to transmit a RREP packet to node **A**, as shown in Figure 4.7(d).

The bordercasting process implements query control mechanisms (Haas & Pearlman 2001) that allow intermediate relay nodes to become aware that the routing zone is being queried, which is referred to as query detection level one (QD1). Other nodes that eavesdrop on the RREQ process can also determine that the routing zone is being queried, and is referred to as query detection level two (QD2).

The QD1 scheme is realised by forcing the intermediate relay nodes in between the source node and the peripheral node to record three fields in a "detected queries table", namely, the RREQ identification number, the initiating source node address, and the last bordercasting node (Haas & Pearlman 2001). Using this scheme, it permits the intermediate relay nodes to determine whether or not it has previously encountered the same RREQ packet. This can occur when a peripheral node forwards the RREQ packet back towards the previous peripheral node. Therefore, if the RREQ was previously encountered, it will discard the RREQ packet to prevent propagation in an infinite loop, which in ZRP is termed "loop-back termination".

In QD2, identification of previous RREQ queries are further extended by providing the ability to determine whether or not it has previously encountered the same RREQ packet to all the nodes within range of the transmitting node (i.e., snooping nodes). In this case, if the RREQ has previously entered the same zone, it will discard the RREQ to reduce unnecessary traffic, which in ZRP is termed "early termination".

Furthermore, the bordercasting process also decrements the TTL of each RREQ packet at each intermediate relay node to control the distance of propagation in terms of hop-count. This is similar to the method of the expanding ring search employed by both DSR and AODV, as discussed in Section 4.6.1.

4.7.2 Selective Bordercasting

It is shown that rather than transmitting the RREQ packet to all the peripheral nodes, "the same coverage can be provided by bordercasting to a properly chosen subset of peripheral nodes" (Haas & Pearlman 2001). This scheme is called selective bordercasting (SBC), and the principle of SBC is to eliminate any redundancy in the

bordercasting mechanism so that the number of RREQ packets transmitted is reduced. This can be accomplished by eliminating individual peripherals nodes from the set of all peripheral nodes, where it is evident that a high degree of overlapping of routing information occurs. In other words, if the routing tables of two peripheral nodes, say **A** and **B**, are also neighbours, then both of these two peripheral nodes will share much of the same information within their respective routing zones. This can be represented as $A \cap B \gg (A \cup B) - (A \cap B)$, given that the intersection (the similar information) between the two routing tables of nodes **A** and **B** is much larger compared to the dissimilar information. This is especially the case as the zone radius R increases. As a result, it becomes unnecessary to bordercast a RREQ packet to both nodes **A** and **B**, as the same zone coverage can be achieved by only transmitting a RREQ packet to one of these nodes.

To determine which individual peripheral nodes may be eliminated from the set of all peripheral nodes, SBC requires an extended IARP awareness region of twice the zone radius, i.e., $R_{SBC} = 2R$ (Haas & Pearlman 1998a). A revised scheme called "distributed bordercast" (DB) that requires an extended IARP awareness region of $R_{DB} = 2R - 1$ has also been proposed by Haas & Pearlman (Haas & Pearlman 2001). According to the SBC algorithm, the initiating source node must determine which are the peripheral nodes within the routing zone defined by R, termed the inner peripheral nodes. Also, the SBC algorithm must determine those peripheral nodes within the routing zone defined by R_{SBC}, termed the outer peripheral nodes. The algorithm then determines the "minimal partitioning subset" of the inner peripheral nodes that covers the outer peripheral nodes, to which the initiating node will bordercast a RREQ packet to. Therefore, the SBC algorithm can still maintain full coverage of the network area by reducing the high degree of overlapping routing zones by eliminating the redundant peripheral nodes.

The algorithm utilised by SBC to obtain the minimal partitioning subset is through application of a greedy approximation algorithm (Haas & Pearlman 2001), that is derived by Johnson (Johnson 1973). The information required by each inner peripheral node to determine the subsequent minimal partitioning subset for bordercasting is carried in the RREQ packet, thus increasing the size of the RREQ

packet. For subsequent bordercasting using this method, the previous outer peripheral nodes become the next inner peripheral nodes, such that these peripheral nodes in common with the subsequent minimal partitioning subset can be eliminated. In this way, the RREQ packets are guided away from the initiating source node, however, care must still be taken with regard to loop-back termination to prevent the RREQ from propagating in an infinite loop.

In a study by Haas & Pearlman (Haas & Pearlman 2001), it is discovered that ZRP is vulnerable to simultaneous query overlap within a time frame called the "bordercast propagation window". This is caused by intermediate relay nodes forwarding the RREQ packet before the query control mechanisms of QD1 and QD2 can detect the transmissions by neighbouring intermediate relay nodes. The solution proposed to resolve this problem was to implement a random back-off delay, termed the random query processing delay (RQPD). In this way, the peripheral nodes wait a random amount of time to allow for any existing RREQ transmissions to arrive near the border of the routing zone. The author suggests an RQPD of 10 ms, to allow the peripheral nodes sufficient time to employ the QD1 and QD2 techniques, thereby giving the peripheral nodes a chance at early termination of the RREQ packet. All things considered, the author concludes that route discovery delay in ZRP is comparable to flood searching, but with less routing overhead (Haas & Pearlman 2001).

It is determined that bordercasting in fact results in an effect similar to flooding, especially without query control (Van Der Werf & Chung 2004). This can be deduced by considering that in order to reach all the peripheral nodes, one must pass through each intermediate relay node in the routing path. This is further intensified if several peripheral nodes have multiple intermediate relay nodes in common.

Furthermore, selective bordercasting requires an awareness region of $R_{SBC} = 2R$ hops, and uses a greedy approximation to minimise the number of overlapping RREQ transmissions. The most significant shortcoming of the selective bordercasting scheme is the four-fold increase in routing overhead required by the proactive component; namely, an increase from $O\left(R^2\right)$ to $O\left((2R)^2\right) = O\left(4R^2\right)$.

Coupled with the use of a greedy approximation algorithm, this inherently yields a non-optimum solution for the minimal partitioning subset.

4.7.3 Route Optimisation and Zone Radius Optimisation

The study by Wu, et al. (Wu et al. 2000) proposes a route optimisation for ZRP that is concerned only with the IERP component, as the IARP component always has access to the optimal routing path. The route optimisation is exploited by eavesdropping in promiscuous mode, such that if a non-peripheral node can acquire a shorter routing path between the source and destination nodes, it will forward the RREQ packet to the alternate peripheral node to get to the destination. The author suggests modifying the routing path containing the intermediate peripheral nodes through substitution, thereby not invaliding the ZRP bordercasting scheme. In this way, once the RREQ packet arrives at the peripheral node that has the destination node within its routing zone, the peripheral node can transmit a RREP back to the initiating source with the shorter routing path.

In works by Baldoni & Beraldi (Baldoni & Beraldi 2001; Beraldi & Baldoni 2003), the authors propose an improvement to the proactive route caching scheme that can be used in ZRP which does not rely on expiry timers, termed the caching zone routing protocol (C-ZRP). The scheme operates by monitoring the activity of routing paths within the routing zone delimited by zone radius R, such that any active paths will be cached regardless of routing update packets not being received. Conversely, stale routing paths are immediately removed using RERR packets similar to DSR. The results show a reduction in the number of route discoveries being required due to the increase in availability of valid routing paths within the zone route cache.

In research by Pearlman & Haas (Pearlman & Haas 1999), the authors focus on determining the optimum zone radius R, such that routing overhead is minimised, and is shown conceptually in Figure 4.8. Through observing ZRP routing traffic itself, the authors propose two schemes; namely, minimum searching, and traffic adaptive. Minimum searching refers to ZRP adjusting the zone radius periodically over time until the overhead traffic is minimised, whereas the traffic adaptive scheme

operates by observing the level of ZRP routing traffic itself. For both schemes, it is found that the amount of routing overhead traffic is increased for both relatively large and small values of R, as ZRP reduces to mainly proactive and reactive behaviour, respectively. Similarly, it is found that high node mobility will predominantly increase routing overhead for both IARP and IERP. However, it is observed that node mobility is independent on the optimal value for R if route usage is frequent.

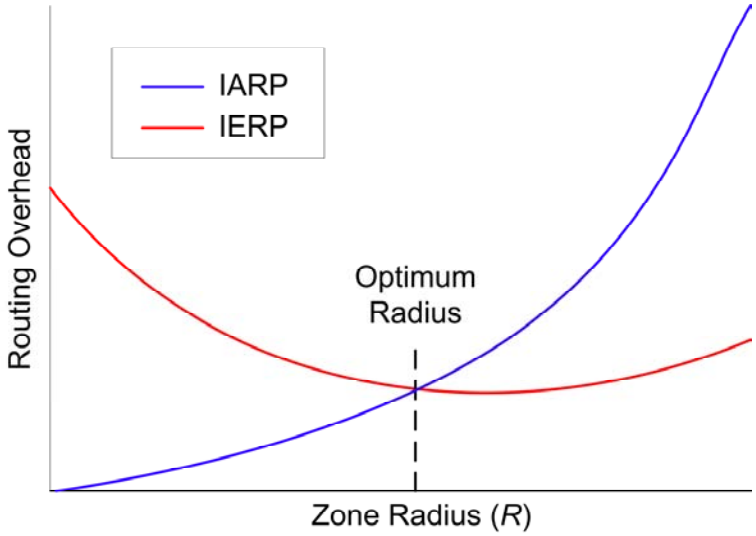


Figure 4.8 – Conceptual trade-off between routing overhead and zone radius showing the dominance of IARP and IERP on either side of the optimum radius line. Thus the optimum zone radius is obtained when the routing overhead ratio between IARP and IERP reaches unity. Graph is adapted from Shea, et al. (Shea, Ives & Tummala 2000).

Succinctly, the authors demonstrate that the optimal zone radius for dense networks with high mobility benefit from small values of R, i.e., $R \ll N$, whereas sparse networks with low mobility benefit from higher values of R, i.e., $R < N$, as expected. Overall, using the minimum searching scheme proposed, the routing zone radius is tuned to within 7% of the minimum overhead, whereas using traffic adaptive the radius is tuned to within 1–2% (Pearlman & Haas 1999).

Meanwhile, Zhang & Jacob (Zhang & Jacob 2003a) studied the impact of having routing zones of varying zone radius for heterogeneous networks, in terms of various

mobility patterns, and propose an adaptive IARP and adaptive IERP scheme. For the IARP component, nodes with high mobility were assigned a small value of R, whereas a higher value for R is assigned for nodes with low mobility. Furthermore, nodes with very high mobility are assigned a zone radius of $R = 0$ to prevent pollution of the IARP routing tables at neighbouring nodes, as these links would be very short-lived. On a related note, the authors of the ZRP framework introduced a revised version called the independent zone routing (IZR) framework (Samar, Pearlman & Haas 2004), which also has support for varying zone radius.

Likewise, the IERP scheme introduces the variable zone radius R as a parameter in RREQ bordercasting. In this way, the intermediate relay nodes that overhear the RREQ packet through QD2 can determine whether or not to participate in the query. Participation occurs when the zone radius specified in the RREQ packet is equal to the intermediate relay nodes value for R, i.e., $R_{RREQ} = R$. Therefore, the quality of the routing path returned in the RREP packet is improved when a high value for R in the RREQ packet is requested, since a high value for R indicates a low mobility routing path.

In a study by Giannoulis, et al. (Giannoulis et al. 2004), the author also considers variable zone radius but with respect to a route failure detection mechanism. The decision to increment or decrement the zone radius R depends on the number of route failures detected by IERP during some time period T. Furthermore, the mechanism attempts to predict the number of future route failures based upon an estimate of the number of nodes outside of the awareness region.

In a study by Wang & Olariu (Wang & Olariu 2004) the authors propose a two-zone routing protocol (TZRP) such that two routing zones are maintained; namely the crisp zone, and the fuzzy zone. The former is used for the proactive IARP component, while the latter is for storing imprecise proactive information. The main purpose of the fuzzy zone is to create a separation between the relation between mobility and routing traffic overhead. This is achieved through accumulation of routing information that would otherwise be discarded by the IARP if the routing information received form a neighbouring node is larger than R. Therefore, before a

route discovery process is initiated, the routing protocol would consult the fuzzy routing table to determine if the destination node is already available. As a result, the total routing overhead can be reduced, however, it is noted that the information within the fuzzy table might not be reliable in terms of the possibility of containing routing loops.

In a work by Yang & Tseng (Yang & Tseng 2005), the authors propose the combination of the ZRP protocol with fisheye state routing (FSR), termed FZRP. In FSR, which is a proactive routing protocol, the accuracy and completeness of the topology information available decreases with respect to distance from the source node. This is of a similar approach to the TZRP fuzzy zone presented by Wang & Olariu (Wang & Olariu 2004). The difference with FZRP is that the frequency of transmitting the update routing information is inversely proportional to the distance from the source node, such that nodes closer to the source are updated more frequently than those near the border of the zone. As such, it is possible to use a comparatively larger value for *R* with FZRP, whilst maintaining a similar IARP update packet size as in ZRP. The key motivation behind this scheme is that as the RREQ packet propagates towards the destination, the IARP information local to each intermediate relay node becomes more accurate. However, it is noted by the authors that the bordercasting process can possibly fail due to the inaccuracies presented in the IARP routing table.

4.8 SUMMARY

In summary, it is seen that several categories of ad-hoc routing protocols exist, including flat routing and hierarchical routing configurations. It is observed that ad-hoc networks have no predetermined topological structure and can exhibit various node mobility profiles, such that nodes are freely able to move into and out of transmission range of one another at any time. The overall goal is to therefore maintain reachability between the nodes to provide stable connectivity between all the other nodes within the network.

An overview of two distinct categories of ad-hoc routing protocols is provided, namely, proactive and reactive routing, which leads on to produce an amalgamation of both called hybrid routing. These categories are then discussed and illustrated by means of an overview of various ad-hoc routing protocols, namely, the destination-sequenced distance-vector (DSDV), optimised link state routing (OLSR), dynamic source routing (DSR), and zone routing protocol (ZRP). The principles learned from examining these routing protocols are then used to formulate the design of a new hybrid ad-hoc routing protocol, which is discussed in Chapter 5.

CHAPTER 5

DESIGN OF A HYBRID ROUTING PROTOCOL (PROACTIVE COMPONENT)

5.1 INTRODUCTION

The proposed hybrid routing protocol, called the Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP), consists of both a proactive and reactive component for use within and outside of the awareness region, respectively. The following two chapters provide an implementation for both the proactive and reactive components for the MultiWARP protocol. In this chapter, the focal point is on designing a new proactive component of a hybrid routing protocol suitable for a multi-hop ad-hoc network. The reactive component is discussed in Chapter 6.

The proactive component relies on periodic routing update packets to be broadcast every μ seconds with a maximum *Time to Live* (TTL) of one, resulting in the packet propagating only to the immediate neighbours of the transmitting node. The aim is to minimise the size of routing packets, and this is achieved by condensing the network topology information into small route update (RUPDT) packets that are transmitted periodically. Furthermore, routing information present in the headers of certain packets traversing the network is exploited through promiscuous listening mode or snooping, such as route reply (RREP) packets. Snooping is when a neighbouring node is not the intended recipient of the packet, yet is able to receive the packet as it is within transmission range. In addition, a novel SuperNode mechanism is discussed that allows certain nodes to carry an extended routing table, thus giving them a larger awareness region.

The proactive routing component is detailed in Section 5.2, and it should be noted that the proactive component described can be interchanged with another proactive scheme without invalidating the MultiWARP protocol. This is due to the design of

the reactive routing component, as detailed in Section 6.2, which utilises the information provided by both components to effectively enhance the operation of the protocol on the whole. In comparison to the ZRP framework, the choices of proactive and reactive components are not explicitly defined for the overall implementation of ZRP (Haas 1997).

5.2 PROACTIVE ROUTING CHARACTERISTICS

As discussed in Sections 3.3.2, 3.5.3, and 4.6, the proactive routing component of MultiWARP is modelled on a combination of distance-vector and path-vector characteristics. Nodes in wireless ad-hoc networks communicate through means of broadcasting, which allows neighbouring nodes to intercept packets within transmission range of the transmitter. As such, there are several ways by which valid routing paths can be extracted and accumulated by a node, either through being the intended recipient or through snooping. With MultiWARP, there are six possible ways of accumulating valid routing paths from received packets, and these are:

i. Data packets that include a Routing Path (RP) header,
ii. Route Update (RUPDT) packets,
iii. Route Request (RREQ) packets,
iv. Route Reply (RREP) packets,
v. SuperNode Request (SNREQ) packets, and
vi. SuperNode Reply (SNREP) packets.

The first two packet types represent the proactive routing component, while the latter four represent the reactive component. The routing path information within the packet headers can be exploited when it is received by a node in promiscuous listening mode (snooping), or is an intermediate relay node responsible for forwarding a packet. This is intuitive since a packet received by an intermediate relay node arrives without solicitation.

5.2.1 Periodic Update Interval

MultiWARP uses a path-vector model for the proactive component, such that any data packet that traverses the network must include a valid routing path to be specified within each packet. In most cases, a route update (RUPDT) packet serves as the primary source of routing path information, when compared to the other five packet types of RP, RREQ, RREP, SNREQ, and SNREP. A RUPDT packet is transmitted periodically by each node every μ seconds through use of a periodic timer. It is assumed that the periodic update interval μ is homogeneous throughout the network by default. The periodic timer is used to activate a new node from its inactive state. The first RUPDT packet transmitted therefore acts as a "hello" beacon to serve as a node start-up announcement for its neighbouring nodes. Usually, an inactive node has an empty routing table, so that the only information contained within its first RUPDT packet will be its own node address.

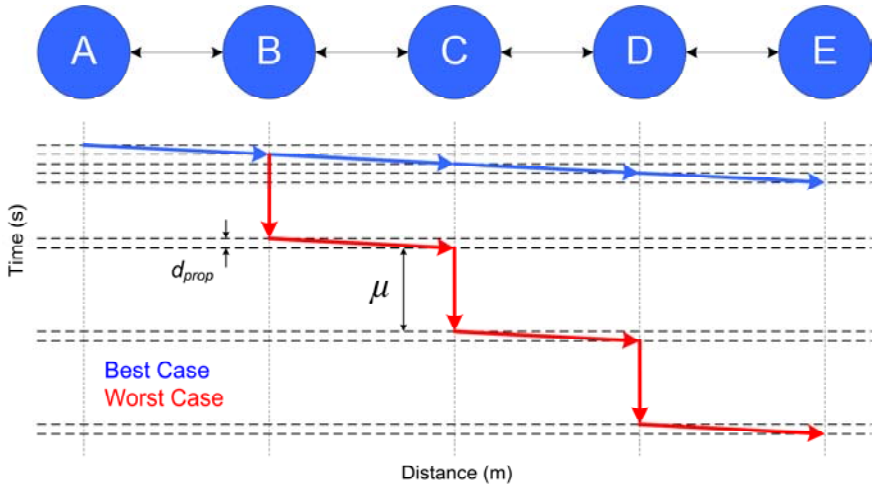


Figure 5.1 – Minimum (best case) and maximum (worst case) time duration of updating the awareness region when a new node is added to an existing network.

Given that the periodic update interval is μ seconds, any new node added to an existing network is automatically self-configured within a maximum of μ seconds after start-up. This is achieved by extracting the routing table information from all the RUPDT packets received from its immediate neighbours within the time frame of

$[0..\mu]$ seconds after start-up. Thus, during this time the new node will become aware of all the nodes within the awareness region. Similarly, the neighbouring nodes within the awareness region must also be made aware of the new node. It follows that all the neighbouring nodes within the $(R-1)$ hop awareness region, centred at the new node, become aware of the new node within the minimum time of $t_{\min} = d_{prop}(R)$ seconds and a maximum time of $t_{\max} = \mu(R-1) + d_{prop}(R)$ seconds, where d_{prop} is the propagation delay between nodes. This is shown in Figure 5.1, where node **A** is the new node, and nodes {B,C,D,E} are its neighbouring nodes. It should be noted that these equations describe the case for a fixed network topology ignoring the effects of mobility, which is investigated in Chapter 7.

5.2.2 Routing Path Snooping

The routing path, also called a source-route, is contained within an additional header called the routing path (RP) header that is encapsulated within the IPv4 packet header. This additional header is provided through the use of an *Options* field, as discussed in Section 2.6. The structure of the RP header is defined in Section A.2. When a data packet is forwarded by the intermediate relay nodes (and ultimately received by the destination node) it becomes possible for each intermediate relay node in between to extract the routing path from the RP header for inclusion in their own local routing tables.

The following example illustrates this process of a node capturing a data packet and extracting the routing path information. Let an intermediate relay node (e.g., node 19) receive a data packet destined for the destination node (node 24) with the following routing path:

$$\{15,16,17,18,19,20,21,22,23,24\}.$$

By observing the routing path in the RP header, node 19 can extract the trailing routing path segment and accumulate it into its own routing table, as follows:

$$\{20,21,22,23,24\}.$$

Similarly, the leading routing path segment can also be added after route reversal, such that:

$$\text{Reverse}\{15,16,17,18\} \Rightarrow \{18,17,16,15\}.$$

It is possible that these extracted routing paths may contain stale routing information. Therefore, care should be taken to prevent these paths from being included into the routing table if these paths are already known to be stale. For instance, if a node or link specified within the extracted information has been recently removed from the local routing table, it could cause the stale node or link to be re-inserted into the local routing table. Thereafter, the stale information will be removed again through a stale node removal algorithm, as discussed in Section 5.2.3, causing rapid routing table oscillation. The problem of rapid removal and re-insertion of node or link information in a routing table could be overcome by using a stale node removal algorithm, which allows a deleted node or link entry to remain in an inactive "expired" state for a period of time.

5.2.3 Stale Node Removal Algorithm

A stale node removal algorithm is used to determine the exact time when a node address or its associated connectivity links should be identified as being invalid, such as when they move out of transmission range. With this algorithm, two parameters are used to detect node inactivity. These are the periodic update interval μ and the associated *Expiry* field within the routing table structure, as detailed in Section 6.6.2. When new routing information arrives at a node by means of a proactive RUPDT packet, the *Expiry* field is refreshed with the local system time of the current node. Consequently, any node or link which has become invalid will no longer have corresponding entries within the RUPDT packet to update them. The node or link is then determined to be invalid or stale when the *Expiry* field shows it has not been updated for some time period. As a result, the stale routing information can be marked as "expired" when the difference between the local system time and the *Expiry* field exceeds a certain time period in order to keep the integrity of the local routing table. Since the expiry time is referenced to the local system time of the current node, a network time synchronisation protocol is not required.

The time period is relative to the periodic update internal μ multiplied by an arbitrary timeout constant C, which is normally set to $C = R$, where R is the awareness region in hops. The node or link is then marked as "expired" when the time given in the *Expiry* field plus the time period becomes less than the local system time of the current node, or in other words $Expiry + \mu(C) < LocalTime$. This allows the receiver to miss a RUPDT packet for up to $\mu(C)$ seconds before it will be marked as "expired". This is referred to as a "slow node death" for a broken node or link as it must timeout for $\mu(C)$ seconds at each node within the awareness region R. This means that a maximum of $\mu(C)(R)$ seconds must elapse before the node or link is marked as "expired" in all routing tables within the awareness region.

To counter this problem, a "fast node death" can be induced by making the time period dependent on the hop-count distance between the invalid node or link and the current node. This can be achieved by decrementing C by one, until $C = 1$, for each hop further away from the current node that the invalid node is located. Using this technique, the timeout period is reduced by μ seconds for each additional hop, and this aids in the expedient removal of the stale node or link. It should be noted that the node or link is not deleted as soon as the node or link is deemed expired. By allowing the entry to remain in an inactive "expired" state for a period of twice the timeout period, $2\mu(C)$, it removes the possibility of a stale node to be re-inserted into the local routing table. The stale node can only be re-activated through reception of a higher sequence number from a RUPDT packet, as discussed in Section 4.4.2, or through the reception of an RP header from the stale node itself.

In MultiWARP, it is possible for routing paths longer than the awareness region to exist within the routing table because of the reactive RREQ and RREP mechanisms. In this case, the RUPDT packet will not contain any information regarding these longer paths as they do not lie within the awareness region R. As a result, for these routing paths the *Expiry* field will not be updated. Instead, for these longer routing paths, the node or link is determined to be invalid after $\mu(X)$ seconds, where X is the total number of hops in the routing path.

5.2.4 Effect of Periodic Update Interval on Stale Node Removal

The periodic update interval μ is the single parameter influencing the proactivity behaviour of the protocol (controlling RUPDT packets), whereas the parameter R influences the reactivity of the protocol (controlling RREQ and RREP packets). As discussed in Section 4.4.1, proactive protocols generally do not perform adequately in high mobility networks, as the routing update packets can fail to keep up with changes in node movements. Thus, for diverse networks that consist of various levels of both low and high node mobility, it becomes advantageous to allow for heterogeneous values of μ to be used by different nodes. This would allow for areas with low nodal mobility to transmit RUPDT packets slower (i.e., a larger μ value), and a faster update interval for areas with high nodal mobility (i.e., a smaller μ value) in order to keep up with topological changes.

The allowing of heterogeneous values of μ to be used in the network conflicts with the algorithm proposed to remove stale nodes, as discussed in Section 5.2.3. This is because the algorithm relies on the local value of μ, and assumes it to be homogeneous throughout the network. Therefore, to provide support for various mobility profiles, the heterogeneous values of μ for the different nodes is included in the *Options* field within the RUPDT header structure and the RUPDT payload structure, and are detailed in Appendix A. This provides the additional information required by the algorithm for the removal of stale nodes in heterogeneous environments. Each node can therefore correctly determine when a node becomes stale due to inactivity, as the algorithm is aware of the rate that each node is transmitting its RUPDT packet.

5.3 CONSTRUCTION OF THE ROUTE UPDATE PACKET

The route update (RUPDT) packet must contain all the information necessary in order to reconstruct the network topology known locally (i.e., all nodes within the awareness region) at a neighbouring node. Thus, the RUPDT packet should contain

information pertaining to all the nodes up to a distance of $(R-1)$ hops away from the source node. The minimum required information (or fields) that must be incorporated into the RUPDT packet include; the node addresses, sequence numbers, and the connectivity between the nodes. Other information such as the heterogeneous periodic update interval or routing metrics (e.g., congestion control information, bandwidth availability, or security authentication) can be specified in the *Options* field for each node entry. It should be noted that before the RUPDT packet is constructed from the local routing table, a route maintenance routine is executed to remove any stale nodes or links, as detailed in Section 5.2.3. Furthermore, "expired" nodes or links are not included in the RUPDT packet.

The RUPDT packet is constructed using a two-stage process in order to minimise the overall packet size. This is achieved by converting the routing path information in the local routing table into short-form, and then condensing this information to create the final encoded RUPDT payload structure. The effect of stream compression and route reduction techniques on routing path information was studied by Van Der Werf & Chung (Van Der Werf & Chung 2005) as a technique to decrease overall RUPDT packet size. However, in practical implementations these two techniques should be rejected due to issues of computational complexity and partial topology reconstruction, respectively. Furthermore, a favourable decrease is RUPDT packet size can be achieved through efficient payload encoding.

The primary two-stage condensation technique for efficient payload encoding is presented subsequently in Sections 5.3.1 and 5.3.2, and is followed by a comparison with two alternative encoding schemes. The difference being that the primary technique supports bi-directional connectivity, whereas the latter two alternative techniques support both bi-directional and uni-directional connectivity at the expense of a larger overall encoding size.

5.3.1 Condensation Stage One

The first stage is to create the *Payload* field of the RUPDT header, as detailed in Section A.3, and shown in Figure A.3. The *Payload* field is comprised of an internal

structure of five fields termed the RUPDT payload structure, as shown in Figure A.4, and contains the topological information needed to depict the $(R-1)$ hop connectivity between the nodes. The RUPDT payload structure is subdivided into $(R-1)$ consecutive *Payload* segments to accommodate all the nodes of a similar hop-count group, ranging from 1 to $(R-1)$ hops.

Each consecutive *Payload* segment has δ node entries, where δ is the value given by the previous segment's *Neighbours* field. For the first segment, this value is given in the *Neighbours* field of the RUPDT header. Each of the δ node entries consist of the first 3 fields of the RUPDT payload structure; namely *Node Address*, *Node Sequence Number*, and the *Options* field (if enabled in the RUPDT header). After the δ node entries are specified, it is then followed by the *Neighbours* field (which is used by the next *Payload* segment to obtain the value of δ), and the current *Payload* segment's *Connectivity Matrix*. This process is illustrated using an example topology with an awareness region limited to $R=5$, as shown in Figure 5.2, where node **A** is the source node constructing the RUPDT packet.

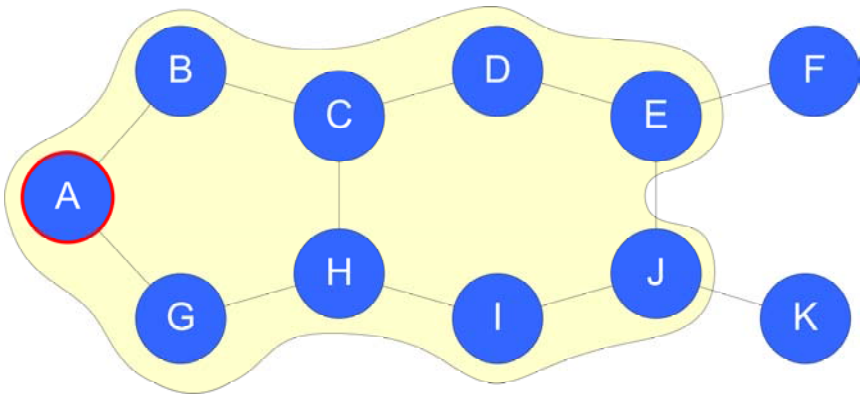


Figure 5.2 – Example topology with an awareness region of *R=5*, showing the (*R–1*) hop connectivity region shaded.

Source node **A** has to create $(R-1)=4$ consecutive *Payload* segments depicting the network connectivity between all the δ nodes of a similar hop-count group, whilst allowing nodes to be specified in multiple segments. From the topology, it is observed that source node **A** can reach the following groups of nodes, given a certain

hop-count, as shown in Table 5.1. The connectivity of the topology is wholly depicted using this table, as nodes reachable using β hops must be connected via nodes reachable using $(\beta - 1)$ hops.

Segment	Reachable Node Addresses	δ Node Entries
Header	{A}	2
#1	{B, G}	2
#2	{C, H}	4
#3	{D, H, C, I}	6
#4	{E, G, I, B, D, J}	0

Table 5.1 – Example topology with an awareness region of *R=5*, subdivided into (*R–1*) segments of similar hop-count groupings.

The information presented in Table 5.1 shows that source node **A** has $\delta = 2$ nodes {**B**, **G**} that are located 1 hop away. Accordingly, the *Source Node Address* field in the RUPDT header is set to node **A**'s address, and the *Neighbours* field is set to 2. Within the RUPDT payload structure, the $(R-1) = 4$ *Payload* segments can then be created from segments #1 through #4, and can be written in short-form as shown in Figure 5.3.

```
A2, BG2, CH4, DHCI6, EGIBDJ0.
```

Figure 5.3 – Short-form representation for the stage one condensation technique (intermediate step 1 of 3).

The connectivity matrix for each *Payload* segment can be created using an $(X \times Y)$ binary matrix, where Y is the value of the previous *Payload* segment's *Neighbours* field, and X is the value of the *Payload* segment's *Neighbours* field previous to Y. It should be noted that the first segment is always (1×1). The connectivity within this matrix is indicated by a single bit, '0' indicating no connection, and '1'

indicating a bi-directional link. The row and column indexes refer to the ordered list of nodes belonging to group Y and the current *Payload* group, respectively.
For segment #1, it is implicit that nodes {**B**, **G**} are connected to source node **A**, thus the (1×2) connectivity matrix will consist of only binary ones, giving:

$$\begin{bmatrix}1 & 1\end{bmatrix}.$$

For segment #2, the nodes reachable within this group are tested for connectivity with the nodes in segment #1. It is observed that node **C** is connected to node **B** but not node **G**, conversely, node **H** is connected to node **G** but not node **B**. The resultant (2×2) connectivity matrix for segment #2 is therefore:

$$\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}.$$

The connectivity matrices for both segments #1 and #2 can be written underneath their respective segments, and is given in short-form in Figure 5.4.

```
A2, BG2, CH4, DHCI6, EGIBDJ0.
    11   10
         01
```

Figure 5.4 – Short-form representation for the stage one condensation technique (intermediate step 2 of 3).

At this point, the nodes in segment #3 are then tested for connectivity with segment #2. It is observed that node **D** is connected to node **C** but not node **H**, node **H** is connected to node **C** and implicitly connected to itself (although denoted as '0'), node **C** is implicitly connected to node **C** and node **H**, and node **I** is connected to node **H** but not node **C**. The resultant (2×4) connectivity matrix is therefore:

$$\begin{bmatrix}1 & 1 & 0 & 0\\ 0 & 0 & 1 & 1\end{bmatrix}.$$

The same process is carried out for segment #4, resulting in the (4×6) connectivity matrix of:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The final RUPDT payload structure for the first stage of condensation, including the node addresses, the value of the number of neighbours, and connectivity matrices (omitting the sequence numbers and optional fields) can then be expressed in short-form as shown in Figure 5.5.

```
A2, BG2, CH4, DHCI6, EGIBDJ0.
    11   10   1100   100000
         01   0011   011000
                     000110
                     000001
```

Figure 5.5 – Short-form representation for the stage one condensation technique (final step 3 of 3).

At this point, the short-form representation is passed to the second stage of condensation in order to reduce the overall size of the expression by removing redundancies.

5.3.2 Condensation Stage Two

The second stage is to reconstruct the topology from the short-form representation of the RUPDT payload structure created in stage one. This is carried out to determine which entries do not provide additional connectivity information, and can thus be omitted from the final RUPDT payload structure. Also, information regarding the bit lengths to be used for encoding the RUPDT payload structure can be determined; namely the *Address_bit_length*, *Neighbour_bit_length*, and *Options_bit_length* fields.

This information is used to minimise the overall RUPDT packet size during final encoding of the RUPDT payload structure before transmission.

Given the short-form representation in Figure 5.5, it is observed that node **A** is the source node with 2 adjacent nodes {**B**, **G**}. From the connectivity matrix associated with nodes {**B**, **G**} in segment #1, it is determined that both nodes are connected to node **A** because of the presence of binary ones, the resultant topology is shown in Figure 5.6.

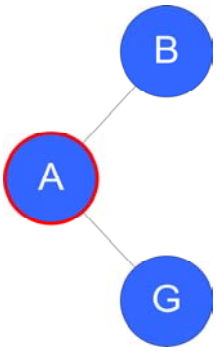


Figure 5.6 – Reconstructed topology from the short-form representation (intermediate step 1 of 4).

The subsequent segment #2 reveals that node **C** connects to node **B** creating a new link between them, but not node **G**. Similarly, node **H** connects to node **G** but not node **B**, giving the following reconstructed topology, as shown in Figure 5.7.

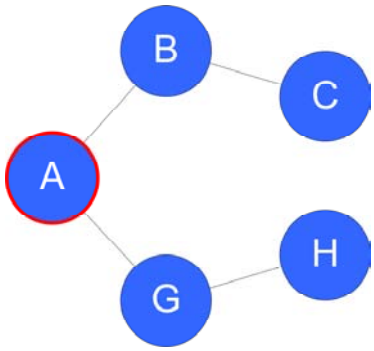


Figure 5.7 – Reconstructed topology from the short-form representation (intermediate step 2 of 4).

With segment #3, all nodes represented create a new link within the topology, except for node **C**. The entry pertaining to node **C** reveals that it is connected to node **H**, a link that had already been created by node **H** by the previous entry. As such, column 3 relating to node **C** in segment #3, and row 3 in the subsequent segment #4 can be omitted from the final RUPDT payload structure. In addition, the previous segment (segment #2) must have its *Neighbours* field decremented by 1 to indicate the removal of the node entry. This yields the modified RUPDT payload structure shown in Figure 5.8, where the omitted column and row is shown by the red line, and the field modification is shown by the underlined blue number. The respective reconstructed topology is given in Figure 5.9.

```
A2, BG2, CH3, DHCI6, EGIBDJ0.
    11   10   1100   100000
              01   0011   011000
                          000110
                          000001
```

Figure 5.8 – Modified short-form representation for the condensation technique (intermediate step 1 of 3).

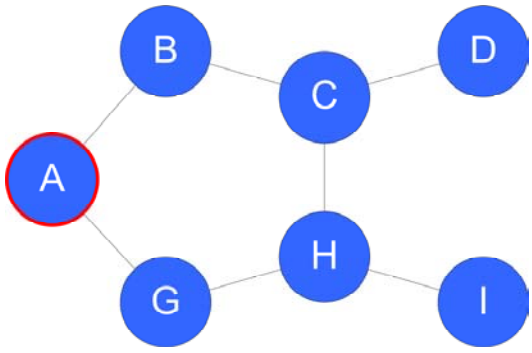


Figure 5.9 – Reconstructed topology from the short-form representation (intermediate step 3 of 4).

In segment #4, only nodes **E** and **J** produce new links, since nodes {**G**, **I**, **B**, **D**} form links that already exist within the reconstructed topology. As such, columns 2 through 4 are removed from segment #4, and the *Neighbours* field in the previous segment (segment #3) is reduced by four, from 6 to 2. This yields the modified

RUPDT payload structure as shown in Figure 5.10, and the respective topology given in Figure 5.11.

```
A2, BG2, CH3, DHCI2, EGIBDJ0.
    11   10   1100   1000000
              01   0011   0110000
                             0000110
                             0000001
```

Figure 5.10 – Modified short-form representation for the condensation technique (intermediate step 2 of 3).

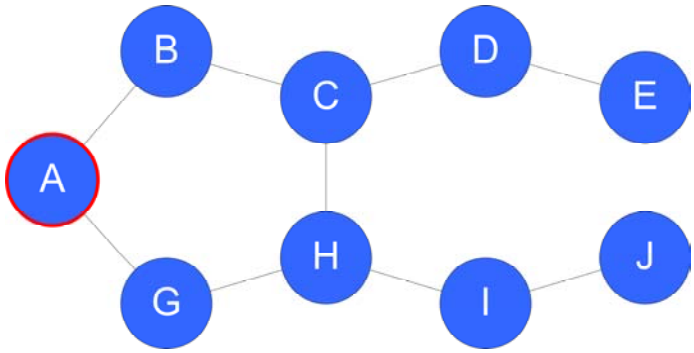


Figure 5.11 – Reconstructed topology from the short-form representation (final step 4 of 4).

The final reconstructed topology given in Figure 5.11 is as expected, and compares identically to the original topology given in Figure 5.2, with the exception of the nodes and connectivity links $\geq R$ hops away being absent. This is because the RUPDT packet contains information for nodes $\leq (R-1)$ hops. The final RUPDT payload in short-form with the redundancies removed is shown in Figure 5.12.

```
A2, BG2, CH3, DHI2, EJ0.
    11   10   110   10
              01   001   00
                         01
```

Figure 5.12 – Modified short-form representation for the condensation technique (final step 3 of 3).

The concluding step in the construction of the RUPDT packet is by encoding the RUPDT header and RUPDT payload structure using the minimum number of bits. In the aforementioned example, the node address length is assumed to be 32-bits as per IPv4, thus the *Address_bit_length* field is assigned the value of 32. From the final RUPDT payload structure, the maximum *Neighbours* field value was 3 (occurring in segment #2) and requires only 2-bits to encode; hence the *Neighbours_bit_length* is assigned the value of 2. Moreover, the *Options* field was not utilised; therefore the *Options_bit_length* field has the value of 0.

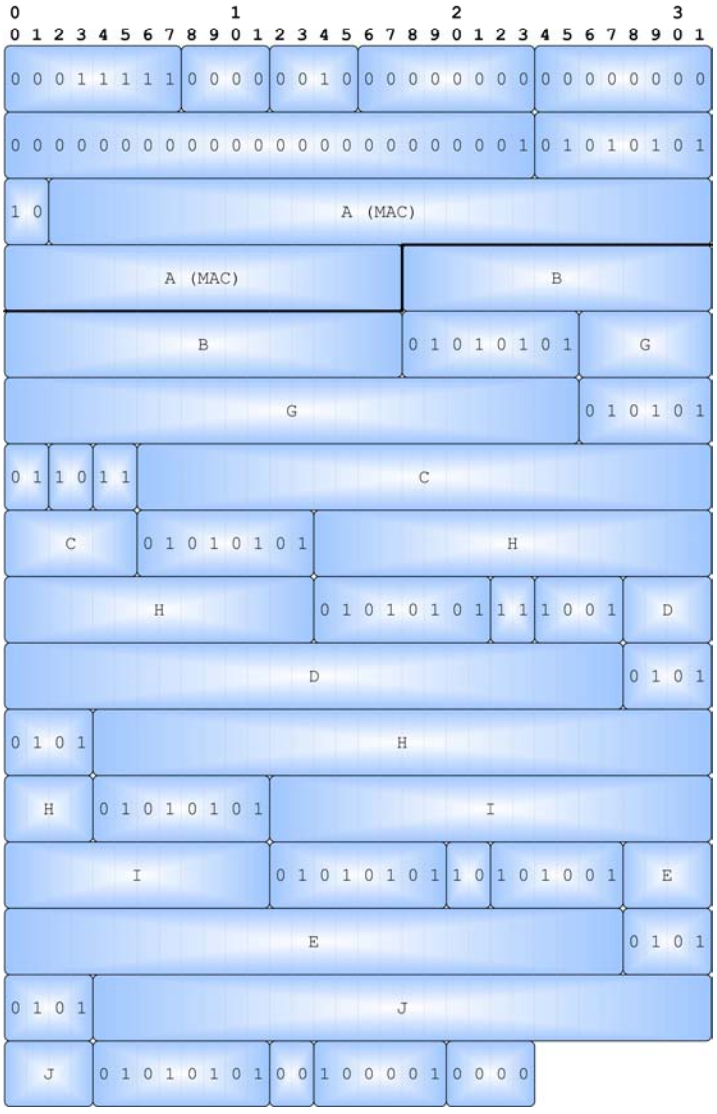


Figure 5.13 – Completed 57 byte RUPDT packet for the final short-form representation. The first 114 bits correspond to the RUPDT header, with the remainder being the RUPDT Payload structure. All *Node Sequence Numbers* are 85 in binary (01010101) for illustration.

During the encoding of the RUPDT payload structure, for each of the δ node entries per segment, the *Node Address*, *Node Sequence Number*, and *Options* field are encoded consecutively using their respective minimum bit lengths. After the δ node entries are specified, it is then followed by the *Neighbours* field, and the connectivity matrix using their respective minimum bit lengths. The *Padding* field at the tail of the completed RUPDT packet is padded with zeros to the nearest byte for alignment purposes with the remainder of the IPv4 packet.

The final encoded RUPDT packet, which consists of both the RUPDT header and the RUPDT payload structure, is presented in Figure 5.13. The overall size of the RUPDT packet required to transmit the $(R-1)$ connectivity region of the topology is compared to two alternative methods, discussed next in Section 5.4.

5.4 ALTERNATIVE ROUTE UPDATE ENCODING SCHEMES

Two alternative encoding schemes are presented and compared to the primary two-stage encoding algorithm, discussed in Section 5.3, that aim to condense the topology information. The *Version* field in the RUPDT header is used to specify which of the three encoding schemes is used, as defined in Section A.3. The first alternative scheme, given in Section 5.4.1, represents the same topological information using a simple $\left(N_{(R-1)} \times N_{(R-1)}\right)$ connectivity matrix, where $N_{(R-1)}$ is the number of known nodes within the $(R-1)$ connectivity region, including itself.

The second alternative scheme, given in Section 5.4.2, is by creating $(R-1)$ segmental $\left(\left(N_{i-1}+N_i\right)\times\left(N_{i-1}+N_i\right)\right)$ connectivity matrices for each of the similar hop-count groupings, where N_i is the value of the *Neighbours* field in *Payload* segment $\#i$. In this scheme, the size of the connectivity matrix created by the first alternative scheme is reduced in size by removing redundancies. For both alternative encoding schemes, the RUPDT header remains identical whilst the RUPDT payload

structure differs. Furthermore, the two alternative schemes presented support both bi-directional and uni-directional connectivity, at the expense of a larger overall RUPDT packet size.

5.4.1 First Alternative Encoding Scheme

For the first alternative scheme, the unordered sequence of *Node Address* fields of all nodes up to $(R-1)$ hops away, denoted by $N_{(R-1)}$ nodes, must be included in the RUPDT payload only once, as shown in Figure 5.14. In addition, their respective *Node Sequence Numbers* and *Options* fields are also included. The total number of node addresses to be included, i.e., $N_{(R-1)}$, is specified by the *Neighbours* field in the RUPDT header. It should be noted that the source node transmitting the RUPDT packet must be included in the unordered sequence of nodes, even though the *Source Node Address* is already specified in the RUPDT header.

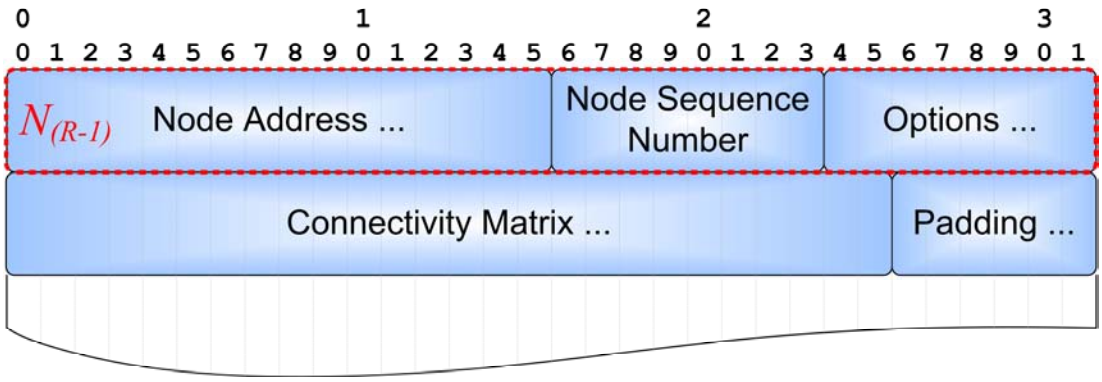


Figure 5.14 – Route Update (RUPDT) Payload structure for the first alternative encoding scheme defined within the RUPDT header.

The column and row index of the connectivity matrix correspond to the sequence of the *Node Address* fields, whereas the entries down the column depict connectivity with the remaining $\left(N_{(R-1)}-1\right)$ nodes. The connectivity is indicated by a single bit, '0' indicating no connection, and '1' indicating a uni-directional link. The ordering of the connectivity matrix is then encoded according to the sequence of bits going

down the column, for each consecutive column. Using the same example topology as shown in Figure 5.2, the short-form representation of the RUPDT payload is shown in Figure 5.15.

```
A9, ABGCHDIEJ.
    111000000
    100100000
    010010000
    001111000
    000100100
    000010010
    000001001
    000000100
```

Figure 5.15 – Example short-form representation for the first alternative encoding scheme.

By comparing the space requirements for encoding the RUPDT packet using this scheme and the one presented in Section 5.3, it is apparent that as $N_{(R-1)}$ increases, the connectivity matrix increases in size proportional to the square. Also, it can be expected that the connectivity matrix will consist of mainly zeros for large values of $N_{(R-1)}$. This is attributed to the inability of many nodes to be connected to the other nodes within the network as they are physically out of transmission range, and therefore are not immediate neighbours with connectivity between them.

5.4.2 Second Alternative Encoding Scheme

The second alternative scheme is to avoid representing the abundant zeros present in the connectivity matrix for the first alternative encoding scheme. This is achieved by creating $(R-1)$ segmental $\left((N_{i-1}+N_i)\times(N_{i-1}+N_i)\right)$ connectivity matrices that depict the connectivity between adjacent segments of similar hop-count, where N_i is the value of the *Neighbours* field in *Payload* segment $\#i$, and N_{i-1} is value of the *Neighbours* field in *Payload* segment $\#(i-1)$. It should be noted that unlike the

scheme presented in Section 5.3, a node is allowed to be a member of only one segment, namely the lowest segment. Using the same example topology as shown in Figure 5.2, it is observed that source node **A** can reach the following groups of nodes, given a certain hop-count, as shown in Table 5.2.

Segment	Reachable Node Addresses	Neighbours
Header	{A}	2
#1	{B, G}	2
#2	{C, H}	2
#3	{D, I}	2
#4	{E, J}	0

Table 5.2 – Example topology subdivided into *(R–1)* segments of similar hop-count groupings, each node must only appear in the lowest segment.

In contrast to the first alternative scheme, the sequence of *Node Address* fields of all nodes up to $(R-1)$ hops, including their respective *Node Sequence Numbers* and *Options* fields, must be ordered according to increasing segment group membership. In addition, the number of nodes in segment #*i*, denoted by N_i nodes where $i = [1..(R-1)]$, must be specified in the *Neighbours* field of the RUPDT header for the first segment, and in the *Neighbours* field of the RUPDT payload, as shown in Figure 5.16, for subsequent segments.

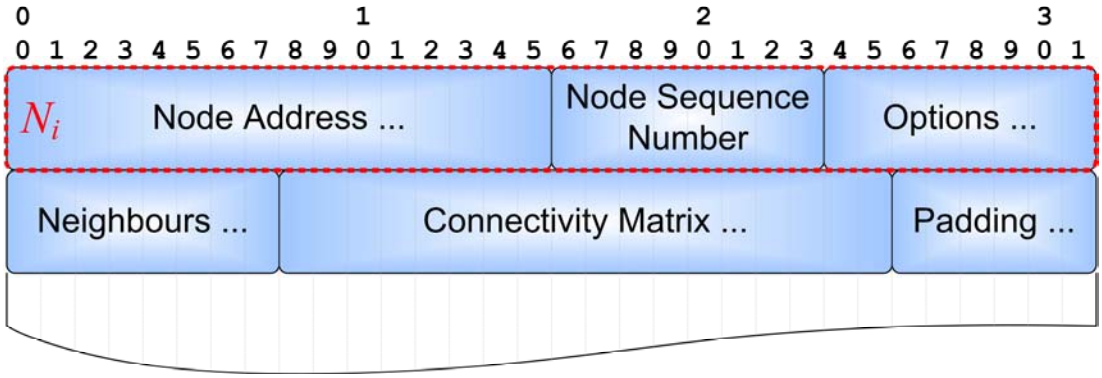


Figure 5.16 – Route Update (RUPDT) Payload structure for the second alternative encoding scheme defined within the RUPDT header.

It should be noted that the internal structure of the RUPDT header for the second alternative encoding scheme is identical to that of RUPDT header as discussed in Section 5.3, the only difference being the interpretation of $\delta = N_i$. The source node transmitting the RUPDT packet must be included in the ordered sequence of nodes as the first node, and also in the connectivity matrix, even though the *Source Node Address* is already specified in the RUPDT header. As a result, the number of nodes contained within segment #1, and hence the first segmental connectivity matrix, consists of $N_0 + N_1$ nodes where $N_0 = 1$.

The segmental connectivity matrix for each *Payload* segment can be created using an $\left(\left(N_{i-1} + N_i\right) \times \left(N_{i-1} + N_i\right)\right)$ binary matrix, where N_i and N_{i-1} are defined previously. The columns and rows of the segmental connectivity matrix then correspond to the sequence of the $\left(N_{i-1} + N_i\right)$ node addresses, whereby the entries in the column depict connectivity with the remaining $\left(N_{i-1} + N_i - 1\right)$ nodes. The connectivity is indicated by a single bit, '0' indicating no connection, and '1' indicating a uni-directional link. The ordering of the connectivity matrix is then encoded according to the sequence of bits going down the column, for each consecutive column. It should be noted that the overlap that occurs between two consecutive segments is not repeated in the payload, as indicated by the shaded areas in Figure 5.17. Again, using the same example topology as shown in Figure 5.2 and using the hop-count groupings given in Table 5.2, the short-form representation of the RUPDT payload is shown in Figure 5.17.

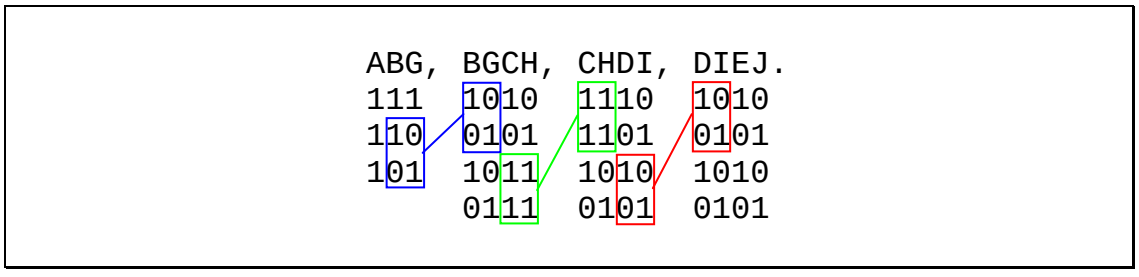


Figure 5.17 – Example short-form representation for the second alternative encoding scheme, showing the redundant overlap that occurs between consecutive segments.

Overall, when comparing the primary encoding scheme, shown in Figure 5.12, it requires only 18 bits to represent the topological connectivity. In contrast, the first alternative encoding scheme, shown in Figure 5.15, requires 64 bits to represent the topological connectivity. The second alternative encoding scheme, shown in Figure 5.17, provides a marginal improvement over the first alternative encoding scheme requiring 45 bits to represent the topological connectivity. Therefore, in this thesis the primary encoding scheme is adopted for use in simulations in order to keep the RUPDT packet size to a minimum.

Lastly, after construction of the RUPDT packet has been completed using one of the three encoding schemes given in Sections 5.3, 5.4.1, or 5.4.2, the packet is disseminated only to the immediate neighbours of the source node. This is achieved by setting the maximum *Time to Live* (TTL) field in the IPv4 header to 1. The packet will travel at most 1 hop before being discarded when the TTL field is decremented to zero. For that reason, the "broadcast storm problem" and its associated flooding problems are avoided, as discussed in Section 4.3.1.

5.5 BROADCASTING OF THE ROUTE UPDATE PACKET

The route update (RUPDT) packet needs to be broadcast periodically according to the periodic update interval μ. The RUPDT packet must be propagated only to the immediate neighbouring nodes of the source node by setting the TTL field. A node will only receive packets which were destined for it (unless it's in snooping mode) by inspecting the *Destination Address* field in the IPv4 header for its unique source address or the common broadcast address. Furthermore, each time a new RUPDT packet is transmitted its sequence number must be incremented, and is discussed in Section 5.5.1.

The RUPDT packet is broadcast by setting the *Destination Address* and *MAC Address* fields of the IPv4 header to the common broadcast IP address of '255.255.255.255' and 'FF:FF:FF:FF:FF:FF', respectively. Similarly, the *MAC Address* field is used to communicate between nodes at the MAC layer level only (i.e., point to point during the RTS-CTS-DATA-ACK exchange) and requires the use

of a protocol to discover the MAC address of the neighbouring node. An example of such a protocol is the Address Resolution Protocol (ARP) (Plummer 1982) which is used to determine the MAC address of node it wishes to communicate with. The use of such a protocol introduces initial transmission delays during the ARP discovery process, and this is employed by the DSR protocol.

In MultiWARP, the use of the ARP protocol becomes redundant as the unique 48-bit MAC address of the source node transmitting a RUPDT packet is included in the RUPDT header structure. As the RUPDT packet is received by all immediate neighbouring nodes surrounding the source node, they become aware of each other's MAC addresses. Given the fact that a MAC protocol communicates with immediate neighbouring nodes only, the MAC protocol is no longer reliant on the ARP protocol for MAC address resolution. The inclusion of the MAC addresses of all nodes within the awareness region in the RUPDT packet is not necessary for the same reason. Each node simply maintains a list of MAC to IP address associations as they are extracted from the RUPDT header. In Section 7.3.4, it is found that there is a 33.6 fold decrease in the time before a data packet can be transmitted due to the removal of ARP when comparing MultiWARP to the DSR routing protocol.

The following example illustrates that only the MAC address of the transmitting source node is required within the RUPDT packet. If node **A** is an immediate neighbour of node **B**, then the MAC addresses of both nodes would have been discovered through the reception of RUPDT packets every periodic update interval μ. However, if a RUPDT packet is not received by node **A** from node **B** (in other words node **B** might be located 4 hops away from node **A**), then node **A** will never try to send an RTS packet to node **B**. As a result, node **A** does not need to know the MAC address of node **B**, as they are not immediate neighbouring nodes.

5.5.1 Sequence Number Algorithm

A sequence number is a number that is assigned to a RUPDT packet, and is used to determine if the information contained within it is newer than a previously received packet. This is used in MultiWARP to determine if a RUPDT packet contains newer

routing path information. This enables the receiving node to determine whether or not to use the routing information provided by its neighbouring nodes, such that only information bearing the highest sequence number is retained, and older information is discarded.

The sequence number in MultiWARP is an 8-bit unsigned integer giving a total of $2^8 = 256$ values in the range of $[0..255]$. The sequence number of 0 is reserved for the initial transmission of a RUPDT packet by a new node, which serves as the node start-up announcement. The sequence number is incremented by 1 before each subsequent transmission of a RUPDT packet, and is circular in nature (excluding the reserved value 0) according to the sequence:

(…, 253, 254, 255, 1, 2, 3, …).

The sequence number is acquired from either the *Source Sequence Number* field of the RUPDT header structure, or the *Node Sequence Number* field of the RUPDT payload structure.

```
Algorithm Determine_Higher_Sequence_Number(unsigned integer S)
{
    Set variable Min equal to the currently known sequence number plus 1
    Set variable Max equal to the currently known sequence number plus 128
    If (S equals '0'), then
    {   Return '2' to indicate first RUPDT packet detected   }
    Else if (Min is greater than Max), then
    {   If (S > Min) OR (S < Max), then
        {   Return '1' to indicate higher sequence number detected   }
    }
    Else if (Min is less than Max), then
    {   If (S > Min) AND (S < Max), then
        {   Return '1' to indicate higher sequence number detected   }
    }
    Else if (S equals the current known sequence number), then
    {   Return '0' to indicate same sequence number detected   }
    Else
    {   Return '-1' to indicate lower sequence number detected   }
}
```

Figure 5.18 – Pseudo-code for determining a higher sequence number.

To determine whether the received RUPDT packet contains a higher sequence number (that is circular in nature), the following algorithm is used. It is described in pseudo-code in Figure 5.18, where S is the sequence number acquired from the received RUPDT packet. Upon execution of the above algorithm, if the returned value is equal to '1' or '2', this indicates a higher sequence number or a node start-up announcement has been received, respectively. Alternatively, if the returned value is equal to '0' or '-1', this indicates the sequence number received is either the same or lower than previously recorded, respectively.

5.5.2 Reception of the Route Update Packet

The reception of a RUPDT packet by the immediate neighbouring nodes of a transmitting source node results in the acquisition of topological information. The topological information contained within the RUPDT payload structure is used to update the local routing table. The topology is reconstructed at the receiving node in a fashion similar to the second stage topology reconstruction illustrated in Section 5.3.2. The difference being the addition of using the *Source Node Address* as the root node connecting the received topology, as given by the RUPDT header. As the RUPDT payload structure contains all the routing path information for up to a distance of $(R-1)$ hops, the inclusion of the root node creates an awareness region covering R hops.

The reconstructed topology created from the RUPDT payload structure is compared to the current routing table available at the local node. The node that transmitted the RUPDT packet incremented its sequence number before transmission, therefore the value of the *Source Sequence Number* in the RUPDT header should always be newer than the currently known sequence number. Given that the sequence number is newer, the information available in the RUPDT packet (such as the *Node Sequence Number* and any metrics) is used to update existing information in the local routing table. In addition, any new information relating to new nodes can be added to the local routing table.

Existing information is defined as any *Node Address* that the receiving node is already aware of. In this case, if the value of the *Node Sequence Number* in the *Payload* segment is bearing a newer sequence number than the currently known sequence number, the newer value is retained and the older value will be discarded. In the case when the *Node Sequence Number* remains the same, and the *Options* field is enabled in the RUPDT header, it proceeds to compare the current and new routing metric values. If the new routing metric is better than the existing metric, the information will also be retained.

During the updating of the routing table at the local node, care should be taken to avoid using routing path information that could possibly include stale node or link information. For example, if a node or link within the routing table was recently deleted, it could possibly cause the stale node or link to be re-inserted into the local routing table. This problem is addressed by allowing the respective node or link entry to remain in an inactive "expired" state for a period of time using the *Expiry* field, as opposed to being immediately deleted, as discussed in Section 5.2.3.

When a RUPDT packet is received from a node indicated by the *Source Node Address* field, the receiver has validated that it can communicate directly with that node. This is because the RUPDT packet is not forwarded due to the maximum TTL of 1. As a result, the *Expiry* field for the node entry corresponding to the *Source Node Address* can be refreshed to the local system time. When a node no longer receives a RUPDT packet from a previously known immediate neighbouring node, or if a RUPDT packet no longer contains information regarding previously known nodes or links, the *Expiry* field will no longer be refreshed. Using this scheme, nodes or links that are not being refreshed will eventually be removed using the stale node removal algorithm, as discussed in Section 5.2.3.

The processing of the received RUPDT packet yields $(R-1)$ consecutive *Payload* segments each containing δ node entries, as discussed in Section 5.3. The initial value of δ is given by the *Neighbours* field of the RUPDT header. Subsequent values of δ are given by the previous *Payload* segment's *Neighbours* field. Each *Node Address* given in the *Payload* segment along with its respective *Node Sequence Number* is compared to the currently known sequence number. Again, if the

sequence number is newer, the new *Node Sequence Number* and the *Expiry* fields are updated for the corresponding node entry accordingly. In addition, the *Connectivity Matrix* that specifies the connectivity between the δ node entries in the current *Payload* segment, and the node entries from the previous *Payload* segment have their respective *Expiry* fields refreshed. It should be noted that the *Expiry* field will not be refreshed if it is currently marked as "expired", as discussed in Sections 5.2.2 and 5.2.3. For new information, the *Node Address* and its respective *Node Sequence Number* from the *Payload* segment is added to the local routing table as is, along with any *Options* fields. The *Expiry* field for the new node and the accompanying *Connectivity Matrix* is refreshed to the local system time.

5.6 SUPERNODE MECHANISM

In this section, a novel mechanism is discussed that allows certain nodes, namely "SuperNodes", to carry an extended routing table, thus giving them a larger awareness region. It is advantageous to utilise different sized awareness regions in cases such as large networks with various subnets of mixed densities and differing mobilities. For example, a University campus library would have a higher density and mobility of mobile nodes than a lecture theatre, which in turn has a higher density and mobility of mobile nodes to a small-office environment. In areas of concentrated node density, e.g., a "hot-spot", an extended routing table will alleviate the need for frequent route request (RREQ) packets from being generated. For example, if a node wishes to communicate with other nodes outside of the normal awareness region *R*, it can query the extended routing table to obtain a valid routing path. The SuperNode mechanism does not require heterogeneous values of *R* to operate in different density networks. Instead, it relies on the same covercasting mechanism used by the route request (RREQ) process to deliver the extended routing table information, as discussed in Chapter 6. This is in contrast to requiring heterogeneous values of μ to be used when dealing with networks of different mobilities, as discussed in Section 5.2.4.

The advantage of a node electing to become a SuperNode can alleviate the delay experienced in the route discovery process. The benefit of the SuperNode is to limit

the distance a RREQ packet traverses the network in order to obtain a corresponding route reply (RREP) packet. As to be discussed in Section 6.2.1, the RREQ packet in MultiWARP does not have to physically arrive at the destination node being sought. As such, an intermediate relay node with a valid routing path to the sought after destination node can reply with a RREP packet back to the requesting source node. Therefore, the intermediate relay node can be up to $(R-1)$ hops away from the destination node and reply with a RREP packet. If one of the intermediate relay nodes is also a SuperNode (and has knowledge of the destination node), the SuperNode can reply with a RREP packet. In effect, this means that the SuperNode can be up to $(2R-1)$ hops away from the destination node, as the extended routing table can increase its awareness region by R hops. This results in a reduction in the route discovery latency. Furthermore, any intermediate relay nodes within the reply path of the RREP packet can also extract the routing information via snooping.

5.6.1 SuperNode Request (SNREQ)

The SuperNode mechanism is realised by covercasting a SuperNode request (SNREQ) packet, as detailed in Section A.7 and shown in Figure A.8, every $\mu \cdot R$ seconds to each of the selected candidate nodes, as determined by the covercasting algorithm proposed in Section 6.3.3. The candidate nodes selected have the property of being the smallest subset of nodes that cover all of the peripheral nodes, yet have the maximum topological separation from the other candidate nodes. As a result, obtaining the routing table information from each of these selected candidate nodes through a SuperNode reply (SNREP) packet would give the largest possible increase in topological information, resulting in a larger awareness region of $(2R-1)$. Through re-use of the covercasting algorithm, the SuperNode mechanism is implemented in MultiWARP with no additional computational complexity.

5.6.2 SuperNode Reply (SNREP)

The SuperNode reply (SNREP) packet, as detailed in Section A.8 and shown in Figure A.9, is essentially a data packet that contains an encapsulated RUPDT packet

within it. The SNREP packet is source-routed to the SuperNode that originated the SNREQ packet through use of the routing path (RP) header. The internal packet structure of the SNREP packet remains the same as the RUPDT packet. The difference is that the encapsulated RUPDT packet has the topological connectivity hop-count limited to R hops, and not $(R-1)$ hops. This results in a larger awareness region of $(2R-1)$ hops for the node that originated the SNREQ packet. Upon reception of the SNREP packet by the requesting SuperNode, the RP header is removed and the data encapsulated within it is processed as though it received an RUPDT packet. The only difference being that nodes up to $(2R-1)$ hops away, as opposed to $(R-1)$ hops, can be included during the second stage topology reconstruction, as illustrated in Section 5.3.2.

5.7 SUMMARY

In summary, the design of the proposed proactive component and the methods it used to accumulate valid routing paths is discussed. This included examination of two proactive packet types, namely, data packets including the routing path (RP) header, and the route update (RUPDT) packet. An extensive analysis of the routing update procedure, including the periodic update interval, awareness region, routing path snooping with route reversal, as well as the development of a stale node removal algorithm has been presented.

Furthermore, the construction of the RUPDT packet is illustrated using a primary two-state condensation technique to efficiently encode the payload, and this is compared to two alternative encoding schemes. Lastly, the mechanism by which the RUPDT packet is broadcast using a novel covercasting scheme without requiring an address resolution protocol (ARP) is discussed. The reception of the RUPDT packet is outlined, which included the use of sequence numbers to prevent stale information from being extracted. The above-mentioned aspects of the proactive component are necessary to support the design of the reactive component, which is discussed next in Chapter 6.

CHAPTER 6

DESIGN OF A HYBRID ROUTING PROTOCOL (REACTIVE COMPONENT)

6.1 INTRODUCTION

The proposed hybrid routing protocol, called the Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP), consists of both a proactive and reactive component for use within and outside of the awareness region, respectively. In this chapter, the focal point is on designing a new reactive component of a hybrid routing protocol suitable for a multi-hop wireless ad-hoc network. The design of the reactive component utilises the routing information provided by the proactive component, as discussed in Chapter 5, in order to enhance the operation of several reactive operations. In particular, the routing information contained within the local routing table is exploited in order to guide and minimise route request (RREQ) packets to reduce routing overhead.

The reactive component aims at minimising the number of RREQ packets transmitted in order to discover a valid routing path. Unlike the DSR routing protocol, discussed in Section 4.6, this is carried out in MultiWARP without the need to flood the network. Instead, for route discovery outside of the awareness region, RREQ packets are transmitted to specifically selected nodes within the awareness region through a process termed "covercasting". The method used to minimally select the nodes to transmit the RREQ packets to is based on the application of the NP-complete Set Covering Problem (Karp 1972). This enables MultiWARP to minimise the route discovery delay (latency). At the same time, the routing overhead is reduced compared to other protocols through use of the covercasting technique. The average computational complexity of the algorithm is presented in Section 6.4, and is shown to be characterised as $O(\log n)$.

6.2 REACTIVE ROUTING CHARACTERISTICS

In MultiWARP, the reactive component is modelled on a path-vector algorithm (source-routing) with the assistance of covercasting to efficiently transmit RREQ packets. Covercasting is the term given to transmitting route request (RREQ) packets to specifically selected nodes within the awareness region, in order to discover a routing path that lies outside of the awareness region. The algorithm to determine which specific nodes to transmit the RREQ packet to is discussed in Section 6.3.

The source-routing based approach implies that the routing path utilised by the node must be entirely specified within the packet header. A favourable consequence of this is that MultiWARP is thereby implicitly capable of providing loop-free routing (Shuaib & Aghvami 2009), as opposed to distance-vector based approaches. When a data packet needs to be transmitted, the source node first determines whether or not a routing path to the desired destination node already exists within the local routing table. If such an entry exists, the routing path can then be inserted directly into the packet's routing path (RP) header, as discussed in Section A.2. If multiple such entries exist, the routing path is chosen according to a route selection algorithm, as discussed in Section 6.6. Alternatively, if a routing path to the desired destination node is not available within the local routing table, i.e., the destination node lies outside of the awareness region; a route discovery process must be deployed. This is achieved by generating a reactive RREQ packet to discover the unknown routing path to the destination node. During this process, the outgoing data packet that caused the route discovery process is temporarily stored in the packet buffer of the routing protocol.

The packet buffer is capable of holding packets that do not have any routing path information determined as yet. The packet buffer must be of sufficient length in order to provide adequate temporary storage of packets during the route discovery process. If the packet buffer becomes full, any packet passed to the routing protocol will be consequently dropped, and thus results in throughput losses and delay disturbances. Whilst the packet is being temporarily held in the packet buffer, the

RREQ process attempts to discover the unknown routing path to the destination node. If a routing path to the intended destination node is obtained through the reception of a route reply (RREP) packet (within the RREQ timeout period), the buffered packet can be released and transmitted towards its desired destination node after the routing path is inserted into the packet's routing path (RP) header. If the RREQ timeout period expires before a corresponding RREP packet is received, the node initiating the RREQ process will endeavour to retransmit the packet some time later, as discussed in Section 2.5.3.

To control the duration of the route discovery process, and thereby allowing the routing protocol to determine whether or not the desired destination node is actually reachable, a RREQ timeout value must be established. The timeout value is specified as the window of time available between the transmission of a RREQ packet, and the reception of a corresponding RREP packet. Firstly, this is achieved by limiting the distance a RREQ packet may traverse in terms of hop-count. This can be achieved by decrementing the time-to-live (TTL) value of a packet, similar to an expanding ring search. Secondly, the RREQ timeout value is then calculated as the amount of time required for the RREQ packet to traverse the maximum distance specified by the TTL, and then multiplying by two to allow for the return of a RREP packet. As discussed in Section 4.6.1, a two-tier expanding ring search with a TTL value of $N/2$ in the first round, where N is the maximum hop-count routing path within the network, and a TTL value of N in the second round, can be used to achieve the best performance. The maximum round-trip time can be calculated using Figure 5.1 to determine the RREQ timeout value, namely the maximum time of $timeout_{\max} = 2\left[\mu(TTL-1) + d_{prop}(TTL)\right]$ seconds, where μ is the periodic update interval and d_{prop} is the propagation delay between nodes.

6.2.1 Transmitting a Route Request (RREQ)

In MultiWARP, the aim of the reactive component of the routing protocol is to minimise the number of route request (RREQ) packets transmitted in order to discover a valid routing path. It is proposed that topological knowledge already

available in the local routing table be applied when determining a routing path to a destination node that lies outside of the awareness region, as defined by radius R. The local routing table created by the proactive component of the protocol contains a great amount of valuable information that can be exploited by the reactive component. A number of assumptions can be made regarding which nodes to ask in order to minimise the number of RREQ packets transmitted, or in other words, which node to transmit the RREQ packet to, as follows:

- If the source node does not know of a routing path to the destination node, its immediately adjacent neighbouring nodes will also most likely not know of a routing path to the destination node.
- It is not desirable to ask two (or more) nodes that are close to each other in terms of hop-count, as they will exhibit a high degree of overlap in their respective routing tables, therefore just select one of the nodes.
- It is not desirable to ask a node that is also reachable using fewer hops, in other words, nodes that can be reached using various shorter routing paths.
- It is not desirable to ask a node that does not act as an uplink to more nodes, i.e., it is a terminating node that has no further connectivity.
- It is not desirable to ask a node where the request has already been propagated towards, or any nodes surrounding where the request came from.
- It is desirable to retain the ability to reach the destination node regardless of its mobility or location within the network.

The first assumption is justified because all immediately adjacent neighbouring nodes are only 1 hop further away from the source node, and therefore the neighbour's knowledge does not broaden the view of the network topology much. This is evident as a high degree of overlapping information occurs in the routing table between neighbouring nodes, as provided by the proactive component. Furthermore, a similar scenario occurs when two (or more) nodes reside mutually close together in terms of hop-count, but are both deemed far away from the source node. In this case, the overlap of information in their respective routing tables remains high, and therefore does not adequately distribute the concentration of topological information evenly. As such, it is sufficient to select just one node to

forward the RREQ packet to, and not both. For that reason, a scheme must be developed in order to determine which nodes to select within the awareness region, termed the "candidate nodes". The method used to minimally select the candidate nodes to transmit the RREQ packets to is achieved through the application of the NP-complete Set Covering Problem (Karp 1972) for a given reachability matrix, as discussed later in Section 6.3.

The RREQ packet is covercasted to the selected candidate nodes through a series of unicast transmissions. The RREQ packet originating from the source node contains the source-route to the selected candidate node within the *Source Route* field of the RREQ header, as shown in Figure A.5. The intermediate relay nodes specified within this field then proceed to forward the RREQ packet towards the candidate node. If the routing path should fail, such that any intermediate relay node is unreachable, the RREQ process can undergo route repair in order to correct the routing path provided in the *Source Route* field, as discussed in Section 6.6.1.

Upon successful arrival at the intended candidate node, the candidate node queries its local routing table to determine whether it is aware of a routing path to the requested destination node, as specified in the *Destination* field. If the candidate node is aware of the destination node, then the routing path from the candidate node to the destination node is appended to the *Source Route* field within the RREQ header. Furthermore, a route reply (RREP) packet, as discussed in Section 6.2.2, can then be transmitted back to the source node. This can be accomplished using a route reversal technique, which consists of reversing the traversed route comprised of intermediate relay nodes back to the source, as discussed in Section 5.2.2. It should be noted that this technique assumes bi-directional links, therefore in order to support uni-directional links, a second approach is by initiating a route discovery process back to the source node.

Alternatively, if the candidate node is unaware of the destination node, the route discovery process is repeated such that the candidate node covercasts the RREQ packet to its selected candidate nodes. Each time appending the routing path to the subsequently selected candidate node to the *Source Route* field. This process

continues until the value of the TTL field of the RREQ packet has been decremented to zero, as discussed previously.

In contrast to the DSR protocol, in which multiple RREQ packets are flooded throughout the network, MultiWARP RREQ packets are targeted at specific nodes in order to reduce the routing overhead. Furthermore, in DSR if two (or more) RREQ packets traverse the network using different paths to the destination node, the second packet will not cause a RREP packet to be returned – even if the routing path found is shorter. This can result in DSR using routing paths that are non-optimal. In MultiWARP, multiple RREP packets can be generated since it is possible that more than one candidate node is aware of the destination node. In this case, the route discovery process always returns the shortest path and possibly several alternative backup routes, if the destination node is indeed reachable.

Moreover, the MultiWARP RREQ packet does not physically have to arrive at the destination node, as per the DSR protocol. In other words, a candidate node up to R hops away from the destination node can reply with a RREP packet which results in minimisation of the route discovery latency. In addition, an intermediate relay node with a valid routing path to the required destination node can also reply with a RREP packet back to the source node, not just the selected candidate nodes. As a RREQ packet is being forwarded closer towards the desired destination node, the candidate nodes have a greater potential for possessing a more up-to-date knowledge of the neighbouring topology, as discussed in Section 6.6.1. Early termination of the forwarding of the RREQ packet and responding with a RREP packet by an intermediate relay node could result in staler information being transmitted back to the source node. Therefore, in MultiWARP the RREQ packet is not terminated, but the intermediate relay node is permitted to respond with a RREP packet.

During the forwarding of the RREQ packet by intermediate relay nodes, neighbouring nodes are able to intercept the packet when in promiscuous listening mode. As such, these nodes are capable of extracting the partially completed routing path to the destination node for inclusion to their local routing tables, as illustrated in Section 5.2.2. Since the neighbouring nodes would be unaware of any impending route repairs possibly required by the intended recipient, routing paths accumulated

from RREQ packets could possibly consist of stale information. As a result, it becomes preferable to snoop on RREP packets, as they will contain a proven valid routing path.

6.2.2 Receiving a Route Reply (RREP)

Upon reception of a RREP packet by the source node that originally initiated the route discovery process, the routing path contained within the *Source Route* field of the RREP packet, as shown in Figure A.6, is used to update the local routing table. The method of extracting the routing path is illustrated in Section 5.2.2, and remains the same for both the intended recipient and any other node snooping the RREP packet in promiscuous listening mode.

For a routing path that spans outside of the awareness region of R hops, the *Expiry* field within the routing table structure is set to the local system time of the current node. This is similar to routes within the awareness region accumulated through the proactive component. However, the *Expiry* field is not updated as no new information is received for that routing path through reception of a RUPDT packet. This is to support the stale node removal algorithm for discovered routes, such that these longer routing paths are determined to become invalid after $\mu(X)$ seconds, where X is the hop-count of the routing path, as discussed in Section 5.2.3. Furthermore, since the RREP packet does not contain sequence numbers for the node addresses comprising the routing path, the existing sequence number information within the local routing table does not need updating.

Since the data packet that triggered the route discovery process was temporarily placed in the packet buffer, the acquisition of a routing path triggers the packet to be released from the packet buffer. The action taken by the routing protocol is then to insert the routing path into the *Source Route* field of the routing path (RP) header, as discussed in Section A.2, and transmit the packet to the intended destination node.

6.2.3 Route Error (RERR)

The routing paths accumulated through the reception of route update (RUPDT) packets and by means of snooping can have a tendency to fail over time, i.e., the routing paths may become invalid. This scenario can occur for multiple reasons, such as sudden removal of a node from the network (e.g., an unexpected link breakage), or if the proactive component has not removed an offending node through means of stale node removal. For example, if the sequence number information has not been disseminated across the network in time. As discussed in Section 5.2.1, this can be attributed to the distance in hops between the source node and the offending node, as there exists a maximum delay of $t_{\max} = \mu(R-1) + d_{prop}(R)$ seconds before the source node becomes aware of the offending node within its awareness region. If the source node decides to use a routing path that includes the offending node, the selected source-route will most likely fail, and therefore require route repair.

A route error (RERR) packet, as shown in Figure A.7, will be generated by the current node if the next-hop in the *Source Route* field of an RP header is unreachable. The RERR packet is transmitted back to the source node that originated the erroneous source-route by reversing the traversed route, indicating the segment where the failure occurred. The current node and all the intermediate relay nodes back to the source node should mark the offending node in their routing tables to the inactive "expired" state, as opposed to the link being immediately removed. The offending node should not be used again in the route selection process until the offending node is re-activated through reception of a new sequence number, as discussed in Sections 5.5.2.

Alternatively, if the offending node was only temporarily unreachable, the reception of an RP header from the offending node can also re-activate the node. However, as discussed previously, the routing path extracted from the RP header could possibly consist of stale information. The RP header should only re-activate the offending node if the RP header was received directly from the offending node, i.e., the offending node is an immediate neighbouring node.

After transmission of the RERR packet, and depending on the *Route Repair* and *Repair Counter* fields, the source-route specified in the *Source Route* field of the RP header can be repaired, as discussed in Section 6.6.1. This is performed in order to prevent the packet from being dropped. If the source-route can be successfully repaired, the packet is forwarded to a different next-hop node that is not the offending node. It should be noted that a RERR packet will still be generated regardless of a data packet being successfully repairable.

6.3 THE REACHABILITY MATRIX

For the proposed reactive component, to determine the routing path to a desired destination node, we must choose which nodes to transmit the route request (RREQ) packets to. The overall objective of the protocol is to minimise the number of RREQ transmissions in order to minimise the route discovery latency, whilst reducing the routing overhead through use of the covercasting technique.

6.3.1 Constructing the Reachability Matrix

To construct the reachability matrix, the first step is to determine the set of nodes which lie 1 hop away from the peripheral nodes, i.e., nodes that are reachable using $(R-1)$ hops. Secondly, this set of nodes is reduced by removing all nodes which can be reached using fewer hops. In other words, we are exclusively choosing those nodes which are reachable using a minimum of $(R-1)$ hops. Lastly, we test the connectedness of each of these nodes to determine if they are terminating nodes that exhibit no further connectivity. This is made possible because the routing table retains routing information for a distance up to the awareness region of R hops that is maintained proactively. As a result, all the nodes that are deemed terminating nodes with no further connectivity are removed from the set. The resultant set of n nodes is termed the “set of candidate nodes”.

The set of candidate nodes are all possible nodes that might know the destination node being sought, or alternatively, are able to further continue the route discovery process through propagation of the RREQ packet. The question that arises is which node (or nodes) should be chosen from the set of candidate nodes to which the RREQ packet should be transmitted, but at the same time retaining a cover of all peripheral nodes. Therefore, it is desirable to select the smallest subset from the set of candidate nodes that cover all of the peripheral nodes; thereby the number of nodes that the RREQ packet must be transmitted to is minimised. This problem can be solved through the application of the NP-complete Set Covering Problem, to be discussed in Section 6.3.2.

Given that all nodes within the network maintain their own routing table, each of the nodes within the set of candidate nodes will have its own topological view of the network. Therefore, each candidate node that lies $(R-1)$ hops from the source node implicitly has knowledge of all the nodes within a radius of R hops from itself, yielding a total awareness radius of $(2R-1)$ hops. Obviously, this knowledge is located $(R-1)$ hops away from the source node, and therefore the source node does not have access to it. Since the source node does not know exactly what each candidate node knows, it can however partially derive this knowledge from its own local routing table point-of-view. In particular, it becomes possible to exploit this information and determine for each candidate node the subset of all the other candidate nodes that it is aware of, i.e., the cross-reachability between candidate nodes.

The set of candidate nodes can be decomposed into n subsets of candidate cross-reachability information. By representing the n subsets in the form of an $(n \times n)$ square matrix, termed the reachability matrix, it becomes possible to apply the proposed algorithm, given in Section 6.3.3, to select the optimum subsets required to yield the minimum set cover. The subsets required in turn correspond to individual candidate nodes to transmit the RREQ packet to. For instance, given a certain network topology with its accompanying routing table, the source node might have a set of 4 candidate nodes {**A**, **B**, **C**, **D**}. For each candidate node within this set, the

algorithm determines if it can reach any of the other candidate nodes using $(R-1)$ hops or less, as derived from the source node's routing table. Assume that each node can inherently reach itself, and additionally that node **A** can reach node **C** and node **D**, node **C** can reach node **A**, and node **D** can reach node **A**. This would produce the reachability matrix M, given in equation 6.1, whereby each consecutive row (or column) represents the cross-reachability subsets of nodes {**A**, **B**, **C**, **D**}, respectively.

$$M = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix} \qquad \text{(Eq. 6.1)}$$

This matrix then denotes the reachability of each candidate node with respect to all other candidate nodes, including itself.

6.3.2 Solving the Reachability Matrix

To solve the reachability matrix, an algorithm is needed to determine which subsets (i.e., rows of the reachability matrix) should be chosen in order to completely cover the set of candidate nodes using the minimum number of subsets. Given a set of n candidate nodes, and the respective n cross-reachability subsets derived from the set of candidate nodes, the $(n \times n)$ square reachability matrix can be constructed as shown in Section 6.3.1. Noting that each candidate can inherently reach itself, the reachability matrix always includes the identity matrix, I. Hence, there always exists the trivial solution of using all subsets as the cover.

The reachability matrix can be solved using an exhaustive search algorithm whereby each possible combination is tested. However, this is clearly undesirable as it will have scaling problems for large n. Another method that can be used is the simple Greedy approximation algorithm (Chvatal 1979). This algorithm works by first choosing the row with maximum degree, and adding that row index to the solution set. Second, all elements from the chosen row in common with any other row is

removed from the matrix. This process continues until the reachability matrix becomes the zero matrix (or alternatively, the union of the accumulated solution set equals the set of candidate nodes), and the result is obtained in the accumulated solution set. The Greedy approximation, however, does not guarantee the optimum solution in terms of minimum subsets required. If the optimum solution requires k subsets, the Greedy approximation will give a solution with at most $k\left[\ln\left(n/k\right)+1\right]$ subsets (Blum & Sleator 2000).

6.3.3 Set Cover Using Proposed Solution

It can be observed from the reachability matrix that much overlapping in candidate cross-reachability occurs between nodes that are topologically close to one another. This overlapping can be exploited in the sense of introducing redundancy yet still retaining a minimum subset solution for cover. The advantage is that the selected candidate nodes can reach a larger number of the remaining candidate nodes, should some of the RREQ packets fail to reach their intended candidate node. In other words, this is desirable because it increases the robustness of the RREQ process.

In the event that the network topology changes due to unexpected link breakages and/or additions during the RREQ process, having a large overlap increases the probability that the discovery process will nevertheless proceed even if certain paths are no longer reachable. During the discovery process, if certain candidate nodes are removed from the awareness region of the covercasting node, the redundancy found within the overlap gives rise to a more robust RREQ process by having an increased probability of still covering all peripheral nodes.

The proposed algorithm guarantees the optimum solution in terms of determining the minimum k subsets required for set cover, whilst exploiting the advantages of the overlap present within the reachability matrix. The algorithm operates according to a breadth-first traversal principle using a dynamically allocated first-in-first-out (FIFO) queue to achieve minimal memory usage, and is given as pseudo-code in Figure 6.1. The data structures and the queuing method used in the algorithm are defined in

Section 6.3.4, and the maximum overlap calculation algorithm is provided in Section 6.3.5.

```
Algorithm Solve_Reachability_Matrix(matrix M)
{
    Create an empty FIFO matrix queue "MatrixQ"
    Create an empty matrix "W" of same dimension as matrix "M"
    Initialise empty set "Accumulated_cover_field" associated with matrix "W"
    Initialise variable "Valid_solution" associated with matrix "W" to false
    Initialise variable "Terminate" to false
    Enqueue matrix M to MatrixQ
    While (MatrixQ is not empty) and (Terminate equals false), do
    {
        Dequeue matrix from MatrixQ into matrix W
        If W is a non-zero matrix, then
        {
            Find all r rows with a '1' element in the first non-zero column, and insert into array R
            For r entries in R, do
            {
                Set temporary matrix T_r equal to W
                Remove all '1' elements from all rows in T_r in common with row index R_r
                Append row index R_r to Accumulated_cover_field associated with T_r
                If T_r is a zero matrix, then
                {
                    Mark T_r as a Valid_solution
                    Calculate number of overlapping nodes given Accumulated_cover_field
                    Set Terminate variable to true
                }
                Else
                {
                    Enqueue matrix T_r to MatrixQ
                }
            }
        }
    }
}
```

Figure 6.1 – Pseudo-code for proposed algorithm to solve the reachability matrix.

Continuing with the example reachability matrix M, given in equation 6.1, the initial dequeued reachability matrix W, given in equation 6.2, can be solved using the proposed algorithm presented in Figure 6.1, as follows:

$$W = \begin{bmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 \end{bmatrix}. \qquad \text{(Eq. 6.2)}$$

Since W is a non-zero matrix, the first non-zero column of W, as indicated by the arrow in equation 6.2, has $r = 3$ elements which are equal to '1' referring to nodes {**A**, **C**, **D**}. The row indexes referring to these nodes are then placed in the array $R_{i=[1..r]}$, yielding $R_1 = \{1\}$, $R_2 = \{3\}$, and $R_3 = \{4\}$, respectively. Subsequently, $r = 3$ temporary matrices termed $T_{j=R_i}$ are created to store the values of the interim calculations. For the $r = 3$ entries in R_i, all elements equal to '1' from all rows in T_j in common with row index R_i (i.e., rows $\{1,3,4\}$) are removed (set difference), as shown in equations 6.3(a–c), respectively.

$$\begin{aligned} T_1 &= W \setminus R_1 \\ &= W \setminus \begin{bmatrix} 1 & 0 & 1 & 1 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \end{aligned} \qquad \text{(Eq. 6.3a)}$$

$$\begin{aligned} T_3 &= W \setminus R_2 \\ &= W \setminus \begin{bmatrix} 1 & 0 & 1 & 0 \end{bmatrix} \\ &= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \qquad \text{(Eq. 6.3b)}$$

$$
\begin{aligned}
T_4 &= W \setminus R_3 \\
&= W \setminus \begin{bmatrix} 1 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \\
&\qquad\quad \uparrow
\end{aligned}
\qquad \text{(Eq. 6.3c)}
$$

For each T_j matrix, the value of R_i is appended to the empty *Accumulated_cover_field* set in order to remember the row indexes chosen in subsequent execution of the algorithm, yielding $\{1\}$, $\{3\}$, and $\{4\}$, respectively. In this instance, none of the T_j matrices are equal to the zero matrix, and therefore all T_j matrices are enqueued and the algorithm is therefore not terminated.

The next iteration of the loop can then be evaluated by dequeuing the head of the matrix queue (which was T_1) into W. Since W is again a non-zero matrix, the first non-zero column, as indicated by the arrow in equation 6.3(a), has $r = 1$ elements which are equal to '1' referring to node {**B**}, and the row index is placed in the array $R_{i=[1..r]} = R_1 = \{2\}$, respectively. Subsequently, $r = 1$ temporary matrices are created to store the values of the interim calculations. In the following discussion, the temporary matrices are shown as $T_{\alpha j=R_i}$, where α refers to the *Accumulated_cover_field* set showing the branching of previous execution, in this case producing T_{12}. At this stage, all elements equal to '1' from all rows in W in common with row index R_1 (i.e., row 2) are removed, as shown in equation 6.4.

$$
\begin{aligned}
T_{12} &= W \setminus R_1 \\
&= W \setminus \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\qquad \text{(Eq. 6.4)}
$$

At this point, the matrix T_{12} has the value of $R_{i=[1..r]} = R_1 = \{2\}$ appended to the *Accumulated_cover_field* set, yielding $\{1, 2\}$. It is observed that T_{12} is a zero-matrix, and therefore the *Valid_solution* field must be marked as true. The maximum overlap, which is discussed later in Section 6.3.5, can be calculated given the entries in the *Accumulated_cover_field* set, as shown in equation 6.5, where $subset_i$ is given by the i entries in the *Accumulated_cover_field* set, and *Len* is the length of the *sum* array.

$$\begin{aligned} overlap &= sum\{subset_1 + subset_2\} - Len \\ &= sum\{[1 \quad 0 \quad 1 \quad 1] + [0 \quad 1 \quad 0 \quad 0]\} - 4 \\ &= sum\{[1 \quad 1 \quad 1 \quad 1]\} - 4 \\ &= 0 \end{aligned} \qquad \text{(Eq. 6.5)}$$

In this case, the maximum overlap is equal to zero, as $subset_2$ (i.e., node **B**) is required for set cover and does not have reachability with any other candidate nodes. Also, node **B** is not reachable from node **A**, therefore node **B** is not an element of $subset_1$ (i.e., node **A**). Finally, the *Terminate* variable is set to true, so that the algorithm does not proceed to calculate a lower level of branching; rather only the current branch is further evaluated for additional solutions.

The algorithm proceeds since the current branch of execution has two matrices remaining in the queue. The next dequeued matrix W (which was T_3) would produce $r = 1$ temporary matrix T_{32} after the set difference operation, given in equation 6.6(a), and enqueue it to the tail of the matrix queue. Similarly, the subsequent dequeued matrix W (which was T_4) would produce the temporary matrix T_{42} after the set difference operation, given in equation 6.6(b), and also enqueue it to the tail of the matrix queue.

$$
\begin{aligned}
T_{32} &= W \setminus R_1 \\
&= W \setminus \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\qquad \text{(Eq. 6.6a)}
$$

$$
\begin{aligned}
T_{42} &= W \setminus R_1 \\
&= W \setminus \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}
\end{aligned}
\qquad \text{(Eq. 6.6b)}
$$

At this point, the algorithm has found no further solutions on the current branch of execution since both remaining matrices in equations 6.6(a–b) are non-zero. As such, the algorithm would terminate finding only one optimum solution using minimum subsets for set cover. The relevant solution is found in the *Accumulated_cover_field* set associated with the T_{12} matrix who's *Valid_solution* is marked as true; namely subsets $\{1, 2\}$ which refer to nodes {**A**, **B**} with a maximum overlap of zero. Therefore, ultimately these two nodes will be used in order to covercast the RREQ packet to.

For completeness, assume that the algorithm does not terminate after finding the optimal solution. The algorithm would dequeue the head of the matrix queue into W, and determine that there are $r = 2$ elements which are equal to '1', and set $R_{i=[1..r]} = \{1, 4\}$ accordingly. The $r = 2$ temporary matrices produced are given in equations 6.7(a–b), respectively.

$$
\begin{aligned}
T_{321} &= W \setminus R_1 \\
&= W \setminus \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\qquad \text{(Eq. 6.7a)}
$$

$$
\begin{aligned}
T_{324} &= W \setminus R_2 \\
&= W \setminus \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\qquad \text{(Eq. 6.7b)}
$$

It is observed that both matrices are the zero-matrix, giving two additional solutions that are non-optimum; namely, using subsets $\{3,2,1\}$ and $\{3,2,4\}$. Furthermore, the second (and final) matrix remaining on the matrix queue will be dequeued into W, and set $r = 2$ and $R_{i=[1..r]} = \{1,3\}$ accordingly. The $r = 2$ temporary matrices produced are given in equations 6.8(a–b), respectively.

$$
\begin{aligned}
T_{421} &= W \setminus R_1 \\
&= W \setminus \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\qquad \text{(Eq. 6.8a)}
$$

$$
\begin{aligned}
T_{423} &= W \setminus R_2 \\
&= W \setminus \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix} \\
&= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}
\end{aligned}
\qquad \text{(Eq. 6.8b)}
$$

Again, it is observed that both matrices are the zero-matrix, thus providing the additional solution of using subsets $\{4,2,1\}$. Note that solution $\{4,2,3\}$ is equivalent to $\{3,2,4\}$ found in equation 6.7b, and is therefore not repeated.

6.3.4 Data Structure Types

The algorithm proposed in Section 6.3.3 makes use of five data structure types, as listed, and are described below:

- MatrixQ,
- matrix,
- Accumulated_cover_field,
- Valid_solution,
- Terminate.

The data structure "MatrixQ" refers to a first-in-first-out (FIFO) queue which is implemented as a linked-list in memory. The advantage of using a linked-list implementation is that memory can be dynamically allocated and released corresponding to the size requirements of the FIFO queue during execution. Accordingly, the enqueue function will cause memory to be dynamically allocated to the tail of the linked-list to store the incoming matrix. Conversely, the dequeue function will return the matrix from the head of the linked-list, and release the dynamically allocated memory used to store it.

The data structure "matrix" refers to an $(n \times n)$ binary matrix, which can be represented as a 2-dimensional array to store the reachability matrix, and subsequent temporary matrices used during execution. An example worst case scenario could consist of a network with 100 candidate nodes; this would require a memory space allocation of $(100 \times 100) = 10000$ bits to store the matrix of binary digits (approximately 1.2 kilobytes). The matrix data structure also has two fields associated with it; namely *Accumulated_cover_field* and *Valid_solution*, in order to

remember the row indexes chosen in subsequent branching, and whether the resultant matrix is a valid solution, respectively.

The data structure “Accumulated_cover_field” refers to an array within each matrix structure in order to store the row index numbers selected during execution. When a matrix has been enqueued for further processing, during the next level of iteration when the matrix is dequeued, the algorithm is aware of the traversal of the previous row selections. This array contains the minimum subset solution for set cover, when the corresponding matrix within the MatrixQ is marked as a valid solution.

The data structure “Valid_solution” refers to a Boolean variable that can take on the value of either true (‘1’) or false (‘0’). This variable is used to determine whether or not a matrix that has been processed on the current level of execution has a valid solution in the associated *Accumulated_cover_field*.

The data structure “Terminate” refers to a Boolean variable that can take on the value of either true (‘1’) or false (‘0’), and is used to terminate the main execution loop when a valid solution is found after the current level of execution has finished.

6.3.5 Calculation of Maximum Overlap

For most cases, multiple valid solutions can be found by the algorithm proposed in Section 6.3.3 that satisfies the minimum subset solution for set cover. For that reason, a second criterion to select the best solution from the set of valid solutions can be used that exploits the overlapping present in the candidate cross-reachability information, as discussed in Sections 6.3.1 and 6.3.3.

The proposed method for calculating maximum overlap, given as pseudo-code in Figure 6.2, is to calculate for each valid solution the overlap that exists between the candidate nodes. If the case arises whereby multiple solutions exist with the same calculated maximum overlap value, a third criterion is used. For example, how many peripheral nodes each of the candidate nodes are aware of, whereby the solution with

the highest number of peripheral nodes is optimum, or alternatively, a solution can be randomly chosen.

```
Algorithm Calculate_Maximum_Overlap(matrix M, Accumulated_cover_field)
{
    Initialise array "Sum" of length Len to zero
    For x = 1 to the number of entries in Accumulated_cover_field[i=[1..x]], do
    {
        Add all elements in row Accumulated_cover_field[x] from matrix M to Sum
    }
    Overlap equals the summation of all elements in Sum, subtract Len
}
```

Figure 6.2 – Pseudo-code for calculating the maximum overlap.

The final chosen solution, therefore, has the advantage that the selected candidate nodes can reach a larger number of the remaining candidate nodes should a RREQ packet fail to reach its intended candidate node. This increases the robustness of the RREQ process should the network topology unexpectedly change by introducing redundancy, yet still retaining a minimum subset solution for cover.

6.4 COMPUTATIONAL COMPLEXITY OF ALGORITHM

The objective is to obtain the computational complexity of the proposed algorithm to solve the $(n \times n)$ reachability matrix. The reachability matrix denotes the reachability of each candidate node with respect to all other candidate nodes. The solution to solve the reachability matrix was presented in section 6.3.2, and determined which candidate nodes to transmit the RREQ packet to using the covercasting technique. It was shown that each candidate can inherently reach itself, thus causing the reachability matrix to always include the identity matrix, I. Furthermore, it is noted that the reachability matrix is diagonally symmetrical along the identity matrix. Using these characteristics, there exist a finite number of valid matrices for any given $(n \times n)$ square reachability matrix.

6.4.1 Reachability Matrix Combinations

The objective is to decompose the diagonally symmetrical $(n \times n)$ matrix into n terms in order to determine the computational complexity of the algorithm. There exist a finite number of valid matrices, and given the diagonally symmetrical property, the reachability matrix can be divided into 2 components, as shown in Figure 6.3.

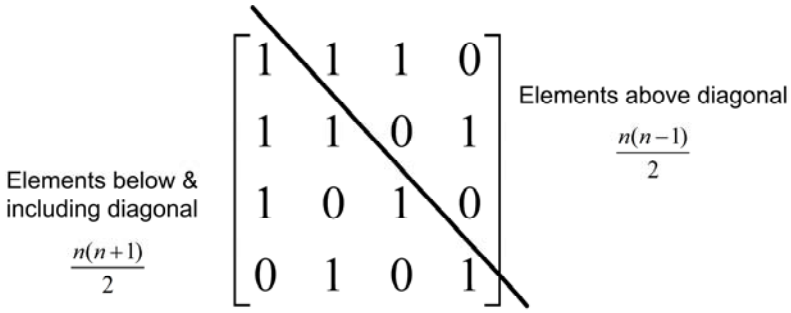


Figure 6.3 – The number of elements in a reachability matrix below and including the main diagonal, and above excluding the main diagonal.

The total number of elements in an $(n \times n)$ matrix is n^2, which can be rewritten as shown in equation 6.9:

$$n^2 = \frac{n(n+1)}{2} + \frac{n(n-1)}{2} \qquad \text{(Eq. 6.9)}$$

where $\frac{n(n+1)}{2}$ represents the number of elements including the main diagonal (the identity matrix), and $\frac{n(n-1)}{2}$ represents the number of elements excluding the main diagonal. By using the diagonally symmetrical property, we can assume that the elements above the main diagonal are equivalent to the elements below the main diagonal, such that element (x, y) is equivalent to element (y, x). As a result, there are only $\frac{n(n-1)}{2}$ bits required to represent all valid combinations of an $(n \times n)$ reachability matrix, such that equation 6.9 can be rewritten as shown in equation 6.10:

$$n^2 = \left[\frac{n(n-1)}{2} + n\right] + \frac{n(n-1)}{2}$$
$$= 2\left[\frac{n(n-1)}{2}\right] + n \qquad \text{(Eq. 6.10)}$$

where $2\left[\frac{n(n-1)}{2}\right]$ represents the number of bits above and below the main diagonal, and n represents the number of elements comprising the main diagonal, as shown in Figure 6.4.

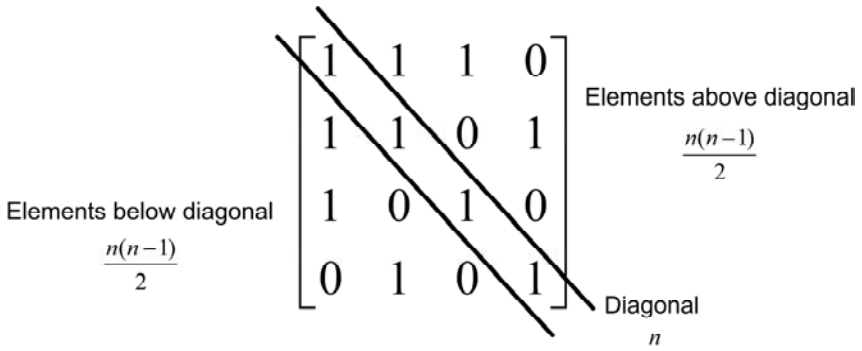


Figure 6.4 – The number of elements in a reachability matrix below the main diagonal, above the main diagonal, and the diagonal itself.

As the reachability matrix is a binary matrix (the elements of the matrix are either '0' or '1'), the term $\frac{n(n-1)}{2}$ represents the exponent of the binary power. The total number of valid reachability matrices, M_{valid}, that can be produced is given by equation 6.11:

$$M_{valid} = 2^{\frac{n(n-1)}{2}} \qquad \text{(Eq. 6.11)}$$

The resultant value of equation 6.11 for a given n yields the number of reachability matrices that are diagonally symmetrical and contain the identity matrix. Listings of valid reachability matrices for $n = \{1, 2, 3, 4\}$ are provided in Appendix B.

In order to express equation 6.11 as a series of n terms, the term $\frac{n(n-1)}{2}$ can be expanded as follows, making use of equation 6.9:

$$\begin{aligned} n^2 &= 1+3+...+(2n-1) \\ &= \sum_{i=1}^{n}(2i-1) \end{aligned} \qquad \text{(Eq. 6.12a)}$$

$$\begin{aligned} \frac{n(n+1)}{2} &= 1+2+3+...+n \\ &= \sum_{i=1}^{n}(i) \end{aligned} \qquad \text{(Eq. 6.12b)}$$

Thus, the expansion for $\frac{n(n-1)}{2}$ can therefore be written as follows:

$$\begin{aligned} \frac{n(n-1)}{2} &= n^2 - \frac{n(n+1)}{2} \\ &= \sum_{i=1}^{n}(2i-1) - \sum_{i=1}^{n}(i) \\ &= \sum_{i=1}^{n}(2i-1-i) \\ &= \sum_{i=1}^{n}(i-1) \end{aligned} \qquad \text{(Eq. 6.12c)}$$

Combining equation 6.11 and equation 6.12c, yields the product given in equation 6.13:

$$\begin{aligned} M_{valid} &= 2^{\frac{n(n-1)}{2}} \\ &= 2^{\sum_{i=1}^{n}(i-1)} \\ &= \prod_{i=1}^{n} 2^{(i-1)} \end{aligned} \qquad \text{(Eq. 6.13)}$$

which is not possible to be decomposed into its individual iterations using a summation in terms in n, without the product operator or the exponentiation base 2 preceding it. Thus, an exhaustive simulation is used to derive the relationship

between how many iterations are required by the proposed algorithm in order to solve the reachability matrices.

6.4.2 Computational Complexity Simulation

The computational complexity depends on the number of iterations (branchings) the proposed algorithm must make in order to arrive at the optimal solution. The number of iterations is dependent on the particular arrangement of the binary elements within the reachability matrix, and is illustrated in Section 6.3.3. An exhaustive computer simulation is used to evaluate the computation complexity by recording the number of iterations, i, required by the proposed algorithm in order to arrive at the optimum solution for every valid $(n \times n)$ reachability matrix.

The number of reachability matrices produced for a particular value of n is given by equation 6.13. The summation of the interim values of each iteration, i, is equal to the value given by equation 6.13 for a given value of n. The results for the values of $n = \{1, 2, 3, 4, 5, 6, 7, 8\}$ are tabulated in Table 6.1. This table shows the occurrence of the number of reachability matrices that require i iterations before being solved by the proposed algorithm.

n	$2^{\frac{n(n-1)}{2}}$	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8
1	1	1							
2	2	1	1						
3	8	4	3	1					
4	64	23	34	6	1				
5	1024	256	627	130	10	1			
6	32768	5319	22946	4137	350	15	1		
7	2097152	209868	1600007	269888	16597	770	21	1	
8	268435456	15912975	215701872	35149292	1619362	50442	1484	28	1

Table 6.1 – The occurrence of the number of reachability matrices that require i iterations before being solved, for values of $n = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

The decomposition graph for the different values of n is shown in Figure 6.5, and shows a consistent relationship between the varying reachability matrix sizes.

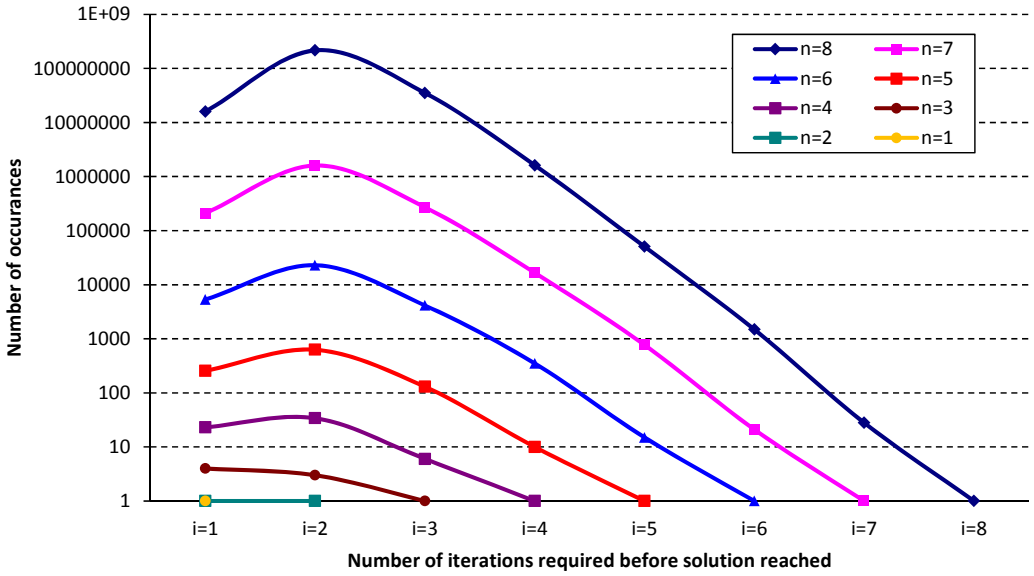


Figure 6.5 – Decomposition graph showing a consistent relationship between the varying reachability matrix sizes, for values of $n = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

n	$\sum_{i=1}^{n}(i \cdot x)$	i=1	i=2	i=3	i=4	i=5	i=6	i=7	i=8	Average
1	1	1								1.00000
2	3	1	2							1.50000
3	13	4	6	3						1.62500
4	113	23	68	18	4					1.76563
5	1945	256	1254	390	40	5				1.89941
6	65103	5319	45892	12411	1400	75	6			1.98679
7	4289917	209868	3200014	809664	66388	3850	126	7		2.04559
8	559503361	15912975	431403744	105447876	6477448	252210	8904	196	8	2.08431

Table 6.2 – The occurrence of the number of reachability matrices that require i iterations before being solved, multiplied by the number of times the event occurred, for values of $n = \{1, 2, 3, 4, 5, 6, 7, 8\}$.

The number of iterations required to solve the matrix must be multiplied by the number of times the event occurred, x, to determine the average, as given in Table 6.1. The results are tabulated in Table 6.2.

The average computational complexity can be determined using the summed number of iterations, divided by the total number of valid combinations of an $(n \times n)$ reachability matrix, as given by equation 6.13. The average computational complexity from this simulation is shown in Figure 6.6.

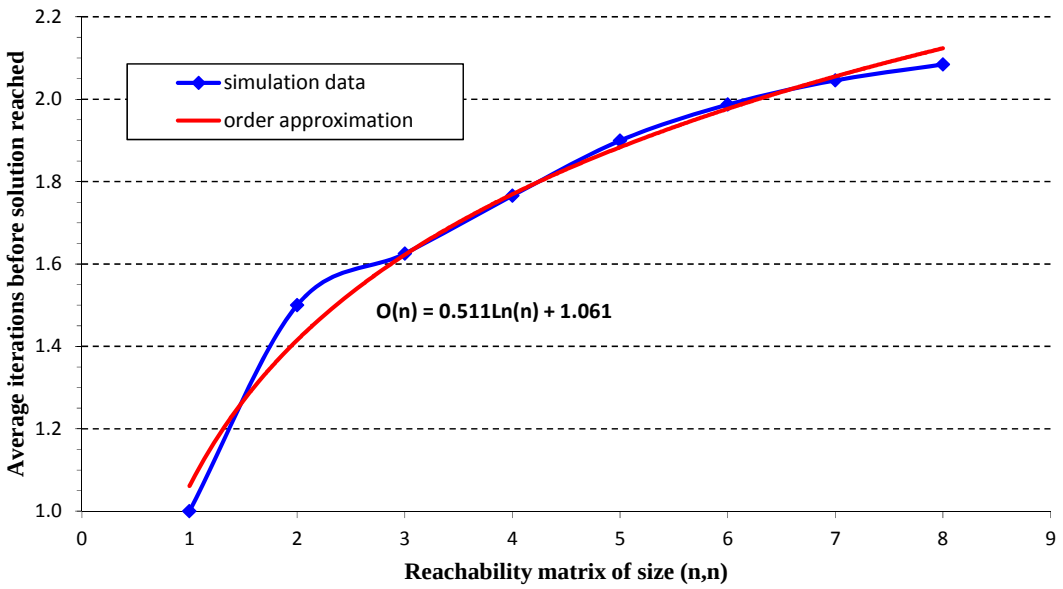


Figure 6.6 – Average computational complexity graph showing the average number of iterations required before a solution is reached depending on the size of the reachability matrix. The overall computational complexity of the proposed algorithm is characterised as $O(\log n)$.

It can be seen that the complexity order equation closely approximates that of $O(n) = 0.5\ln(n) + 1$. The overall complexity of the proposed algorithm is therefore characterised as $O(\log n)$. Accordingly, the proposed algorithm has good scaling qualities for networks with a large number of nodes, i.e., when large values of n are used to solve the $(n \times n)$ reachability matrix. An extensive comparison between many proactive, reactive, power-aware, and hybrid routing protocols including their

computational complexities is given in appendices A – D in a publication by Arslan, et al. (Arslan, Chen & Di Benedetto 2006).

6.5 CONTROLLING PROPAGATION

In Section 6.2.1, an assumption was made that it is not desirable to ask a node where the RREQ has already been propagated towards, or any nodes surrounding where the request came from. To adhere to this assumption, a scheme for controlling the propagation of RREQ packets must be devised. One method is to encapsulate the complete list of addresses of all the nodes that have been covered into the RREQ packet. This allows the selected candidate nodes that receive the RREQ packet to remove the relevant nodes from its set of candidate nodes before any subsequent RREQ packet is transmitted. The cost of this method prohibits its use, as the inclusion of the many node addresses that have been covered greatly increases the RREQ packet size.

A method in which the RREQ packet size remains small is by using a similar technique discussed for constructing the reachability matrix in Section 6.3.1. This is whereby a node can partially derive the routing table knowledge of any node it is aware of from its own point-of-view. Using this method, it becomes favourable to include only the set of candidate nodes in the RREQ packet, as opposed to the addresses of all the nodes that have been covered. The complete list of all the addresses that have already been covered by the RREQ process can then be derived from the local routing table of the selected candidate nodes' point-of-view. In this case, when a candidate node receives a RREQ packet, it is able to determine the partial topology known by the source node or the previous candidate node. Using this information, it becomes possible to remove the nodes in common with the current set of candidate nodes that lie near the previous candidate node. Therefore, there is no requirement to specify the complete list of all the node addresses in the RREQ packet that have been previously covered by the RREQ process. Instead, only the set of candidate nodes of the previous candidate node is explicitly required, as the remaining nodes will be implicitly removed through exploitation of the topological knowledge already available in the local routing table.

6.5.1 Forking-Node Removal Problem

One problem that can arise from the above scheme for controlling propagation is when multiple valid solutions are found by the maximum overlap algorithm given in Section 6.3.5. In the case when two (or more) solutions exist, and thus only one is selected, this can cause the RREQ process to become topologically divided similar to a fork. This is due to the complete removal of all nodes in common which are needed in order to propagate the RREQ packet further, thereby cutting off any possibility of reaching the other half of the network. This is related to the assumption that it is not desirable to ask two (or more) nodes that are close to each other in terms of hop-count, as they will exhibit a high degree of overlap in their respective routing tables. An example network topology showing this behaviour is given in Figure 6.7, and this scenario is termed the "forking-node removal problem".

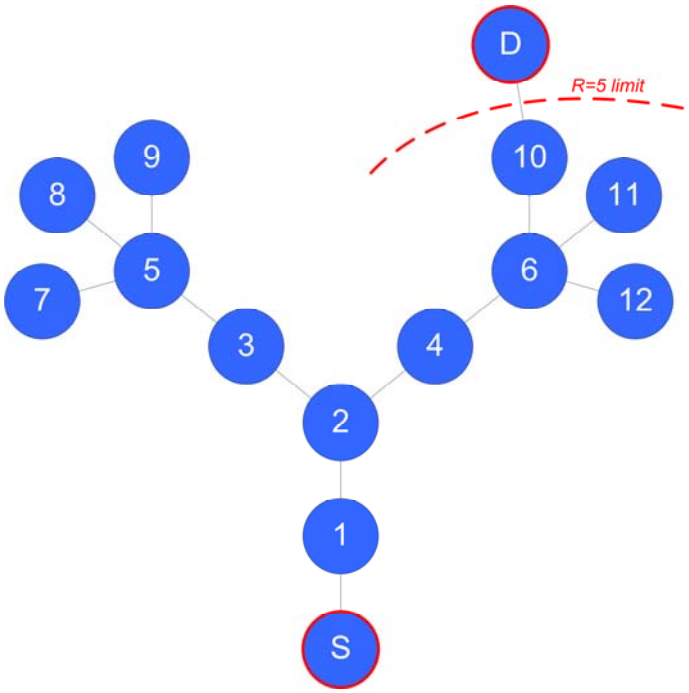


Figure 6.7 – Example network topology with *R=5* exhibiting the forking-node removal problem.

The network topology for node **S**, given in Figure 6.7, has an awareness region of $R = 5$ and two candidate nodes {**5**, **6**} that cover all the peripheral nodes {**7**, **8**, **9**, **10**, **11**, **12**}. It is observed that candidate node **5** can reach candidate node **6** using 4

hops, which is less than R. The reachability matrix for this topology is given by equation 6.14, using the procedure discussed in Section 6.3.1.

$$M = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix} \qquad \text{(Eq. 6.14)}$$

This matrix implies that mutual cross-reachability exists between the two candidate nodes. The forking-node removal problem arises because the optimal solution for minimal set cover requires only one subset, either $subset_1$ corresponding to node **5,** or $subset_2$ corresponding to node **6,** as the maximum overlap for both subsets are equal. If $subset_1$ is chosen, the source node that initiated the RREQ process will never find the destination node **D**. This is because the RREQ packet transmitted to node **5** will indicate that node **6** has been previously covered by the RREQ process, as the set of candidate nodes {**5**, **6**} is encapsulated within the RREQ packet. Therefore, even though node **5** is topologically less than R hops away from node **6**, it cannot reach node **6** because node **5** will remove node **6** from its set of its candidate nodes due to the reachability between the source node and node **6**. In effect, the source node instructs node **5** to remove node **6** from its set of candidate nodes, as the source node itself has selected node **6** as a candidate node.

The forking-node removal problem can be easily detected and resolved. The solution is to explicitly remove the address of node **6** from the RREQ packet destined for node **5**, and vice-versa. Therefore, if node $subset_1$ is chosen by the source node, node **5** can still covercast a RREQ packet to node **6** thereby retaining the ability to reach the destination node. This solution therefore preserves the minimum number of RREQ packets to be transmitted from the source node. On the contrary, if the mutual cross-reachability was removed, whereby the source node would modify the reachability matrix M into a (2×2) identity matrix, this would result in the source node covercasting a RREQ packet to both candidate nodes. This would increase the number of RREQ packets transmitted by the node exhibiting the forking-node removal problem. However, it could result in lower route discovery latency as the 4 hop traversal between node **5** and node **6** (or vice-versa) would be avoided.

6.5.2 Example Protocol Operation

Given the topology in Figure 6.8, the goal is to determine the routing path between the source node and the destination node, indicated by node **S** and node **D**, respectively.

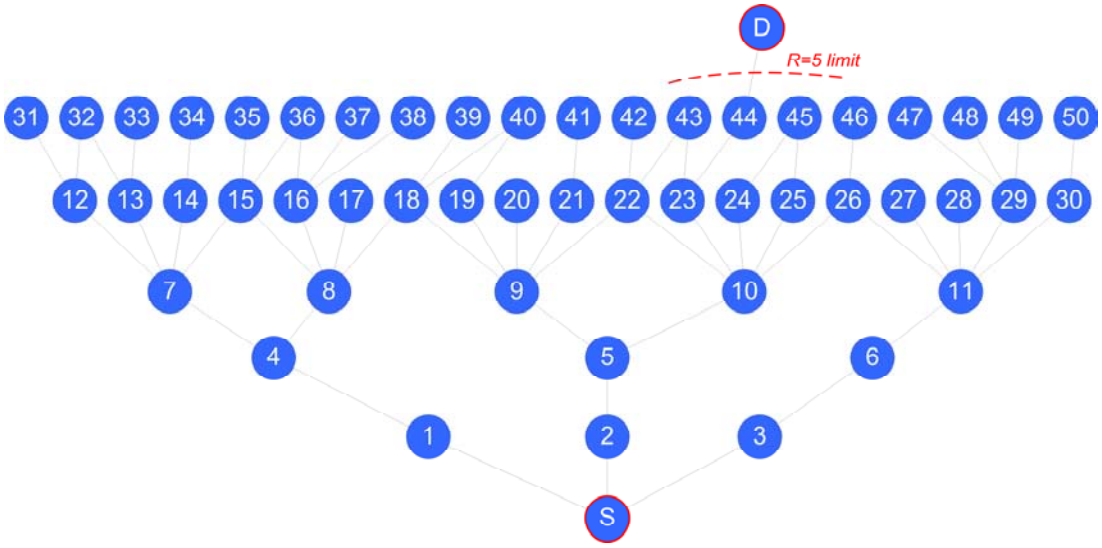


Figure 6.8 – Example network topology consisting of 52 nodes with *R=5*.

In this example, the awareness region has a radius of $R=5$ hops, and the network exhibits a total of 52 nodes. This includes 20 peripheral nodes and 19 nodes that are reachable using a minimum of $R-1$ hops, and is denoted by the $n=19$ set of nodes as follows:

$$(R-1) \text{ nodes} = \{12,13,14,15,16,17,18,19,20,21,22,23,24,25,26,27,28,29,30\}.$$

By observing that the four nodes $\{17,20,27,28\}$ are terminating nodes with no further connectivity, they are removed from the set, yielding an $n=15$ set of candidate nodes:

$$\text{set of candidate nodes} = \{12,13,14,15,16,18,19,21,22,23,24,25,26,29,30\}.$$

The set of candidate nodes can be decomposed into $n = 15$ subsets of candidate cross-reachability information. Given that each node within the network has an awareness region of $R = 5$, it is observed that candidate node **12** can reach candidate nodes **13**, **14** and **15** within 2 hops via node **7**. Similarly, candidate node **12** can also reach candidate nodes **16** and **18** within 4 hops via path "**7–15–8**" or "**7–4–8**", noting that node **17** is removed as stated previously. Additional candidate nodes cannot be reached within the constraint of $R \leq 5$. Furthermore, each candidate node can always reach itself, therefore the subset for each candidate node always includes itself. Combining these results, the candidate cross-reachability information for node **12** is given as follows:

$$\text{cross-reachability for node 12} = \{12, 13, 14, 15, 16, 18\}.$$

Performing this operation for the remaining 14 candidate nodes results in the following reachability matrix, given by equation 6.15, whereby each consecutive row (or column) represents the cross-reachability subsets of the set of candidate nodes, respectively.

$$M = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{(Eq. 6.15)}$$

The above matrix ***M*** can be solved using the algorithm to solve the reachability matrix given in Section 6.3.3, and using the maximum overlap algorithm given in Section 6.3.5. After application of these two algorithms, the minimum number of subsets required for coverage is 2, and multiple solutions exist. The output of the algorithm selects subsets 6 and 9, which have a maximum overlap of 10 candidate nodes. Subsets 6 and 9 refer to nodes **18** and **22** respectively, and both nodes have knowledge of 2 bordering nodes. Thus, the source node **S** covercasts the RREQ packet to nodes **18** and **22**, for which routes already exist. Upon reception of the RREQ packet at these selected nodes, the same procedure is executed. Node **22** checks to see if it knows node **D**, which it does, and replies with a RREP packet. This RREP packet contains the reversed path that the RREQ packet took as it traversed the network, and the appended path between node **22** and node **D**, as follows:

```
NODE 22 FOUND ROUTE from S to D:
--> S, 2, 5, 9, 22, 10, 23, 44, D.
```

At this point node **S** will have obtained a valid route to node **D**. This can be locally optimised further at node **S** by using the routing table information to reduce the hop-count by removing the redundant section (9, 22), yielding:

```
--> S, 2, 5, 10, 23, 44, D. (optimised)
```

However, node **18** also checks to see if it knows node **D**, which it does not, and proceeds to build its own cover matrix whilst taking into account the forking-node removal problem. The cover matrix for node **18** is given in equation 6.16, where the subset rows refer to nodes **12**, **13**, **14**, **23**, **24**, **25**, and **26** respectively.

$$M = \begin{bmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix} \qquad \text{(Eq. 6.16)}$$

Again, the minimum number of subsets required for coverage is 2, and multiple solutions exist. However, all solutions exhibit zero overlap, thus the selection criteria is now dependent on the number of bordering nodes that each candidate node knows. From the routing table, it is indicated that nodes **12**, **13** and **26** have knowledge of 2 bordering nodes, with the remainder having knowledge of only 1. Thus the optimum solution is reduced to only two options – subsets 1 and 7, or subsets 2 and 7. The outcome is randomly chosen with a probability of 50%. Assuming subsets 1 and 7 are chosen, which refers to nodes **12** and **26** respectively, the RREQ packet will now be covercast to these nodes. This will result in the following outcome:

```
NODE 12 HAS NO CANDIDATES. Dead End.
NODE 26 FOUND ROUTE from S to D:
--> S, 2, 5, 9, 18, 9, 22, 10, 26, 10, 23, 44, D.
```

It can be seen that two small loops exist within the returned RREP path. However, they can be efficiently removed using two iterations:

```
NODE 26 FOUND ROUTE from S to D:
--> S, 2, 5, 9, 22, 10, 26, 10, 23, 44, D. (#1)
--> S, 2, 5, 9, 22, 10, 23, 44, D. (#2)
```

It should be noted that these loops can also be removed at the RREQ initiators as they occur, or at the initiator of the RREP packet, as opposed to removing them at the source node. Again, the path can be locally optimised further at node **S** by using the routing table information to remove the redundant section (9, 22), yielding:

```
NODE 26 FOUND ROUTE from S to D:
--> S, 2, 5, 10, 23, 44, D. (optimised)
```

For completeness, if subsets 2 and 7 were chosen, the discovery process would result in the same outcome. It should be noted that for the purposes of this example, any intermediate relay node that knew of a valid routing path to the destination node did not reply with a RREP packet.

6.6 ROUTE SELECTION

When a data packet needs to be transmitted, the source node must first determine whether or not a routing path to the desired destination node exists. This is achieved by querying the local routing table for a valid routing path to the desired destination node, as discussed in Section 6.2. If a valid routing path is found, this path is inserted into the packet's routing path (RP) header as detailed in Section A.2. If the case arises whereby multiple routing paths exist between the source node and the destination node within the local routing table, a routing path chosen is according to a route selection algorithm. Otherwise, if a routing path to the destination node is not available, i.e., the destination node lies outside of the awareness region; a route discovery process must be deployed, as discussed previously in Section 6.2.1.

Many route selection algorithms can be devised, and their overall behaviour depends largely on the selection criteria used. By using selection criteria, it becomes possible to rank the multiple valid routing paths according to a single or composite metric and then choosing the path with the highest rank. A composite metric can be based upon metrics such as the hop-count of the routing path, time since uplink sequence number reception ("uplink integrity"), first in sorted order, or at-random. Alternatively, statistics such as lifetime and stability (Zhang et al. 2010), bit-error-rate (BER) (Ferrari & Tonguz 2007), repair-ability, congestion status, bandwidth availability, or the usage statistics of the routing path can also be used (Gouda & Schneider 2003).

The hop-count metric is used as the primary selection criterion to obtain the shortest routing path. The effect of using a secondary selection criterion can dramatically alter the behaviour of network usage. A good candidate for a secondary selection criterion is the uplink integrity method, whereby the time since receiving a new sequence number from an immediate neighbour is used. This is a measure of how recently that immediate neighbour has successfully communicated with the current node. Since each node transmits a RUPDT packet every μ seconds, the current node should receive a higher sequence number from its immediate neighbour within a time frame of $[0..\mu]$ seconds. The ranking process then sorts the valid routing paths according to the uplink node with the most recent arrival of its respective

RUPDT packet bearing a higher sequence number. Alternatively, the at-random method disregards the uplink integrity method using sequence numbers. With this method, the uplink node is randomly chosen from the set of all possible uplink nodes available for all the valid routing paths leading to the destination node.

The effectiveness and performance of these two secondary route selection criteria, namely, the uplink integrity method and the at-random method, are evaluated in Section 7.3.6. Moreover, the MultiWARP protocol is flexible by allowing other route selection algorithms to be implemented through use of the *Options* field within the RUPDT header and RUPDT payload structure, as discussed in Sections A.3 and A.3.1, respectively. By using the *Options* field, any additional metric information can be transmitted for each node such that it can be utilised during the ranking process of the alternative algorithm.

6.6.1 Route Repair and Route Optimisation

The source-route specified in the *Source Route* field of the RP header can become invalid due to various network conditions, such as the mobility of nodes causing unexpected link breakages. As a consequence, if a data packet containing an invalid source-route cannot be successfully delivered to its uplink node, it is consequently dropped. In MultiWARP, to prevent the packet from being dropped, the source-route can undergo route repair in order to correct the routing path specified in the *Source Route* field. Additionally, a routing path can also be optimised on-the-fly if a shorter path can be found. Both of these changes are realised using the information available within the local routing table. The same route selection algorithm is utilised for both the initial routing path retrieval as well as repairing or optimising routing paths.

The availability of route repair and route optimisation depends on the *Route Repair* and *Repair Counter* fields of the RP header, as shown in Figure A.2. The 2-bit *Route Repair* field indicates if the source-route is allowed to be repaired and/or optimised, and is defined in Table 6.3. If route repair is allowed, it is carried out only when a packet cannot be successfully delivered to the next uplink node. The next uplink node is determined to be unreachable by the absence of a clear-to-send (CTS) or

acknowledgement (ACK) packet being received by the MAC layer. In this case, the MAC layer will trigger the routing protocol to transmit a route error (RERR) packet back to the originating source node indicating which segment failed, as discussed in Section 6.2.3. It should be noted that route repair and route optimisation is then activated in order to select a new valid routing path through use of the route selection algorithm, whilst taking care to avoid the offending node. This generally does not cause any additional routing overhead as multiple backup routes are contained within the routing table, and the RERR packet is only 11 bytes in length for IPv4 addressing.

Route Repair	Description
00	Variable source-route, allow repair, allow optimisation.
01	Fixed source-route, allow repair, disallow optimisation.
10	Fixed source-route, disallow repair, disallow optimisation.
11	Reserved.

Table 6.3 – Route Repair field binary patterns and its corresponding description.

As discussed in Section 6.2.3, the offending node is only re-activated through the arrival of a higher sequence number by the reception of a RUPDT packet, or by the reception of an RP header directly from the offending node. In contrast, if the packet is successfully received by the uplink node, i.e., the arrival of an ACK packet was received by the MAC layer, no further action is taken with regard to route repair. If route repair is not allowed, then the data packet will be dropped regardless of another routing path being available.

During the process of repairing the broken routing path, the path already taken by the data packet to the current node is affixed in front of the new routing path selected, with attention given to prevent circular loops. This repaired routing path is then inserted into the *Source Route* field of the RP header, and the 3-bit *Repair Counter* field in the RP header is incremented by one. This field is an account of the number of times the routing path has been repaired and/or modified, and is required in order to prevent infinite route repairs from occurring. The value is limited to $\left(2^3 - 1\right) = 7$ repairs before the packet is dropped and a RERR packet is generated. Finally, if the source-route was successfully repaired, the packet is forwarded to the new uplink

node. It should be noted that a RERR packet will still be generated nevertheless to indicate the failed routing segment to the source node.

If route optimisation is allowed, the source-route specified in the *Source Route* field of the RP header is checked by the current node to determine if any shorter path can be found. This is done to decrease the overall time it takes for a data packet to reach the destination node by reducing the number of intermediate relay nodes it has to traverse. Since each node transmits a RUPDT packet every μ seconds, it can take up to $t_{\max} = \mu(\lambda - 1) + d_{prop}(\lambda)$ seconds before a node λ hops away becomes aware of the presence of alternative routing paths, as discussed in Section 5.2. Therefore, as a packet is being forwarded towards the destination node, the surrounding intermediate relay nodes have the potential for having a more up-to-date knowledge of the neighbouring topology. For example, given a scenario whereby the current node, say intermediate relay node **I3**, has received a data packet from node **I2** that was originated by node **S** and destined for node **D**, with a source-route as follows:

$$\{S, I_1, I_2, I_3, I_4, I_5, D\}.$$

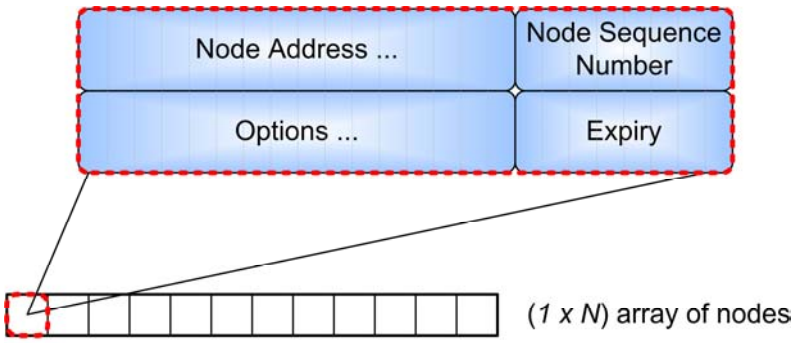
Assume that node **I3** is aware of a shorter path to node **D** than the currently specified path segment $\{..., I_4, I_5, D\}$, for instance, node **I3** and node **I5** have become immediate neighbours due to the mobility of the nodes. As a result, the segment $\{..., I_4, I_5, D\}$ can be replaced by the shorter segment $\{..., I_5, D\}$, yielding an overall saving of 1 hop, namely:

$$\{S, I_1, I_2, I_3, I_5, D\}.$$

The optimised routing path is then inserted into the *Source Route* field of the RP header, and the 3-bit *Repair Counter* field in the RP header is incremented by one, as discussed previously. Subsequently, the packet is forwarded towards the destination node using the new optimised routing path, namely, via node **I5**.

6.6.2 Routing Table Structure

The routing table structure that is stored within memory can be implemented in a number of ways, e.g., a graph, a linked-list structure, or a flat structure; and does not affect the operational performance of the protocol. The implementation chosen is internal to the node, as it is hidden from the other nodes, and thus can be different among nodes within the network. The only requirement is that the data structure of the node entries within the routing table must contain at least the following four fields, which are listed below:



1	1	1	0	0	0	1	1	0	0	0	0
1	1	1	0	1	0	0	0	0	0	0	0
1	1	1	1	0	0	0	0	0	0	0	0
0	0	1	1	1	0	0	0	0	1	1	0
0	1	0	1	1	0	0	0	0	1	1	0
0	0	0	0	0	1	0	0	1	0	0	0
1	0	0	0	0	0	1	1	0	0	0	0
1	0	0	0	0	0	1	1	1	0	0	0
0	0	0	0	0	1	0	1	1	1	0	0
0	0	0	1	1	0	0	0	1	1	1	1
0	0	0	1	1	0	0	0	0	1	1	1
0	0	0	0	0	0	0	0	0	1	1	1

(*N x N*) binary connectivity matrix

i. Node Address (256 bits)
ii. Node Sequence Number (8 bits)
iii. Options (256 bits)
iv. Expiry (64 bits)

Figure 6.9 – Example routing table structure consisting of *N=12* nodes.

The first three fields are defined in Section A.3, with the 64-bit *Expiry* field being used to store the local system time of the current node. The programming method used to represent the connectivity between these node entries is independent, and can be implemented using any technique as it does not affect the protocols operation. For example, a trivial flat structure can be comprised of two parts; an array of

$(1 \times N)$ nodes containing the node entry fields described above, and an $(N \times N)$ binary matrix depicting the connectivity between these nodes is sufficient to meet the minimal requirements, as shown in Figure 6.9.

6.7 SUMMARY

In summary, the design of the proposed reactive component exploits the information contained within the local routing table obtained using the proactive component. This information is used to minimise the routing overhead traffic by selecting specific nodes to propagate the route request (RREQ) packet to, using a novel covercasting mechanism. The construction of the reachability matrix is developed to determine which specific nodes are selected, by exploiting the information available within the awareness region of each node.

Furthermore, an efficient algorithm is presented and illustrated to solve the reachability matrix by calculating the optimum solution for covercasting. The performance and computational complexity of the algorithm was examined through computer simulation and theoretical analysis, and is characterised as $O(\log n)$. Also, a method to calculate the maximum overlap and control propagation is developed to decrease the number of retransmissions and increase the robustness of route discovery process. Moreover, after a route reply (RREP) has been acquired, various route selection criteria are provided which can distribute the network traffic in bursts or evenly among its neighbouring nodes. Lastly, techniques to repair and optimise routing paths on-the-fly are discussed, including a method to assist with the removal of stale routing paths using route error (RERR) packets to maintain the integrity of the routing table.

CHAPTER 7

SIMULATION AND PERFORMANCE RESULTS

7.1 INTRODUCTION

The performance of the proposed routing protocol, called the Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP), is evaluated by computer simulation and is presented in Section 7.2. The objective of the simulations is to ascertain how the proposed routing protocol responds to changes within the network topology, and how it delivers data packets. The performance analysis and evaluations have been carried out under different traffic scenarios using a 50-node random waypoint network topology, as described in Section 7.2.1.

The performance results are presented in Section 7.3 in terms of overall network throughput, packet losses, delay characteristics, route discovery latency and overhead, and routing stability. In Section 7.3.6, a route selection scheme adopted by the proposed routing protocol is described. According to the route selection criteria selected, it either distributes the network traffic in bursts or evenly among its neighbouring nodes (Van Der Werf & Chung 2005). Lastly in Section 7.3.8, multiple alternative routing paths are utilised by the routing protocol to bypass any link-failures, without having to deploy a route discovery procedure.

7.2 SIMULATION ENVIRONMENT

The performance of the proposed MultiWARP routing protocol has been evaluated with the NS-2 network simulator (NS-2 2005) using various network topologies and operating scenarios. The results obtained are then compared with other published routing protocols such as DSR, discussed in Section 4.6, through comparison of results found in other literature and by direct simulation of DSR using NS-2. The OSI layer 2 medium access control (MAC) protocol implemented within the NS-2

simulator closely follows the IEEE 802.11b Wi-Fi standard (IEEE Std 802.11 1999). Similarly, the OSI layer 1 physical layer is implemented in NS-2 as a virtual layer that allows each node to determine whether the signal strength of a transmission is sufficient to provide an acceptable receive quality. Both the physical (PHY) and medium access control (MAC) layers are described in Sections 2.4 and 2.5, respectively.

7.2.1 Simulation Topologies

The simplest network topology is that of a 2-node fixed network, as shown in Figure 7.1. This topology is used to principally demonstrate the correct workings of the protocol in terms of transmitting the correct routing packet types in a timely manner. Also, the 2-node fixed network topology (and the 8-node chain network topology defined hereafter) is used to establish a baseline reference, in terms of throughput and delay characteristics, for use in comparing different network performance.

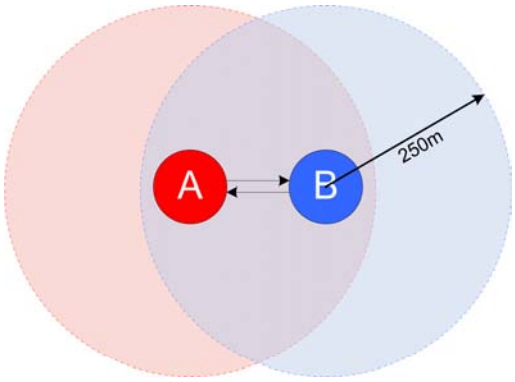


Figure 7.1 – A simple 2-node fixed network topology, with both nodes in transmission range of each other.

The simple 2-node fixed network can be extended into an 8-node fixed chain network, as shown in Figure 7.2, to allow the more complex operations for both the MAC protocol and the routing protocol to be studied. It is noted that the stability of any routing protocol in OSI layer 3 is directly impacted upon by the MAC protocol in OSI layer 2. Later, in Section 7.3.8, it will be shown that certain traffic sources, such as TCP data, can adversely affect the operation of the MAC layer resulting in link breakages (Awdeh 2007). In this case, the MAC layer will report a link-failure

to the routing protocol, causing some routing protocols to initiate routing discovery procedures to find alternative routing paths. Furthermore, it will be shown that MultiWARP can effectively counter-act this TCP instability problem (Xu & Saadawi 2001) in MAC protocol behaviour by utilising alternative backup routes.

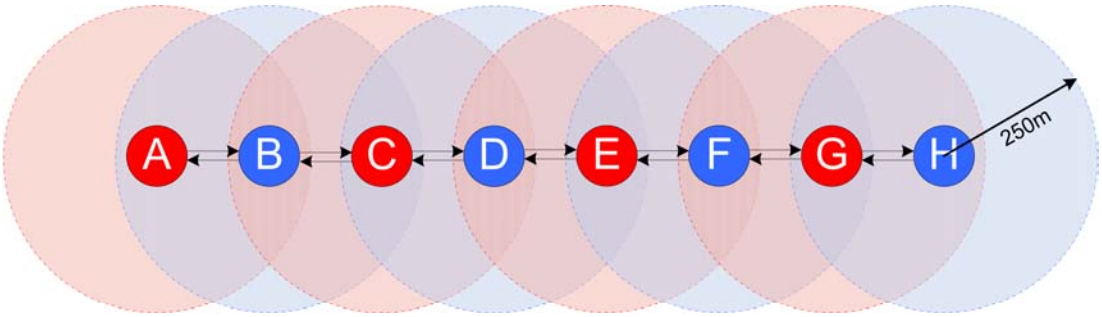


Figure 7.2 – An 8-node chain network topology, with each node being within transmission range of its respective adjacent neighbouring nodes.

The fixed topologies of the 2-node and 8-node chain networks can be made to resemble a more practical network by introducing additional factors. For example, mobility of individual nodes and by extending the topology to 50 nodes randomly distributed within a simulation grid, as shown in Figure 7.3.

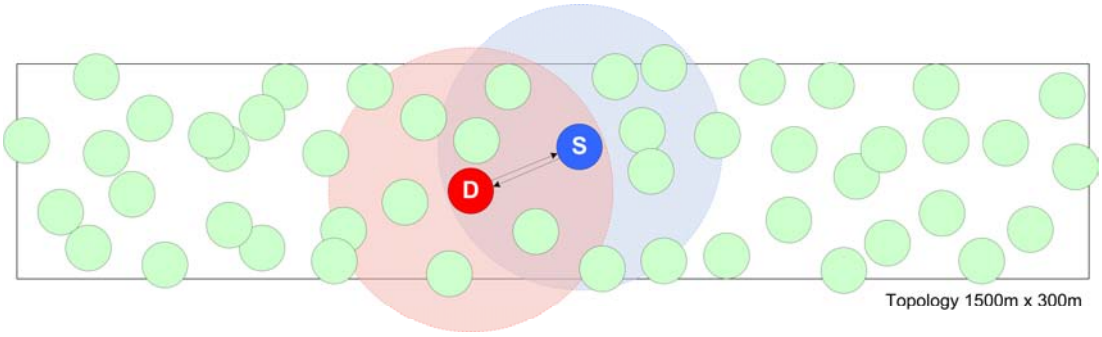


Figure 7.3 – A 50-node random waypoint network topology, with each node being within the transmission range of its respective adjacent neighbouring nodes, indicated by the transmission range circles for source node **S** and destination node **D**.

In an attempt to be able to compare directly with published results, the simulation parameters adopted in this thesis are kept as close as possible to those used in the

literature (Broch et al. 1998; Yoon, Liu & Noble 2003). For example, the use of a 1500m x 300m grid to enclose the 50 nodes is chosen "to force the use of longer routes between nodes than would occur in a square space with equal node density" (Broch et al. 1998). The use of a larger simulation grid, e.g., 1500m x 1500m, was not used in this thesis because most nodes remain unreachable during the simulation time due to an absence of neighbouring nodes. In particular, for a 1500m x 300m grid there is already an average of 303 destination unreachables during the simulation time, discussed later in Section 7.2.3. As a result, using a 1500m x 300m grid will place more 'stress' on the routing protocol as the chance of data traffic flowing using valid routing paths is increased due to less destination unreachables occuring. The simulations used to evaluate the performance of the proposed routing protocol have the network parameters set according to Table 7.1.

Parameter	Value
Network Region Area Size	1500×300 m^2
Number of Nodes (N)	2, 8, or 50 nodes
Transmission Range	250 m
Velocity to Random Waypoint (v)	0, or [1..19] ms^{-1}
Pause Time (p)	0 s
RUPDT Timer (μ)	5 s
Awareness Region (R)	[1..5] hops
MAC Protocol	IEEE 802.11b
Device Bandwidth	2 Mbps
Antenna Position Above Ground	1.5 m
Wireless Physical Layer Frequency	2.412 GHz
Simulation Time	900 s

Table 7.1 – Parameters used in the NS-2 simulation of the 2, 8, and 50 node network.

7.2.2 Simulation Mobility Model

In this research, mobility is introduced in terms of giving each node a random waypoint (RWP) model of movement, as proposed by Johnson & Maltz (Johnson & Maltz 1996) and Broch, et al. (Broch et al. 1998). The movement pattern for each node is assumed independent with a random initial starting point, (x_i, y_i), within the simulation grid, and a uniformly distributed random destination point, (x_d, y_d). A node moves from its initial starting point towards the destination point at a uniformly distributed velocity v in a specified range of between $[v_{\min}, v_{\max}]$. Upon arrival at the destination, the node pauses for p seconds where $p_{\min} \le p \le p_{\max}$. After this pause time, the movement process is repeated with a new destination and velocity. This is carried out repeatedly for each individual node for the duration of a complete simulation run.

According to Bettstetter, et al. (Bettstetter, Resta & Santi 2003), the random waypoint movement model has a non-uniform spatial distribution with nodes tending to cross the centre of the simulation grid with relatively high frequency. As such, it does not truly represent random distributed movements within the network grid. In a study by Yoon, et al. (Yoon, Liu & Noble 2003), the authors suggest that the random waypoint model may not be suitable for simulating ad hoc mobile environments due to the progressive reduction in the average nodal speed over time. Consequently, a node may never reach its destination point during the simulation time, and therefore is no longer considered to actively participate in random waypoint mobility. Over time, these slow moving nodes will build up in number thus causing the average nodal speed of the entire network to decrease. A solution suggested by Yoon, et al. (Yoon, Liu & Noble 2003) is to always assign nodes a non-zero velocity, i.e., $v_{\min} > 0$, and a pause time of $p = 0$. The improvement of using a uniformly distributed velocity of [1,19] ms^{-1} compared to (0,20] ms^{-1} is shown in Figure 7.4.

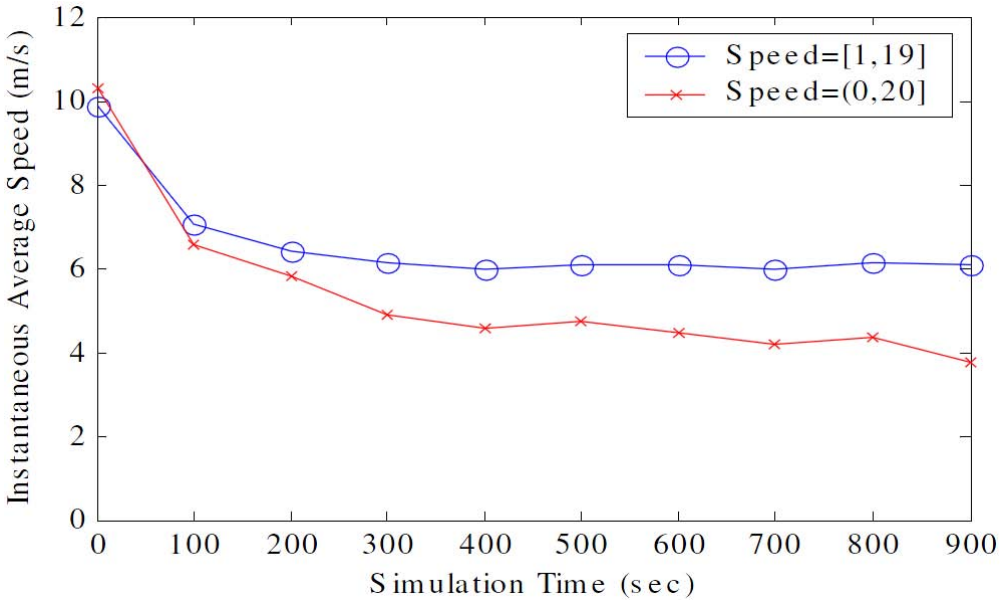


Figure 7.4 – Average speed of nodes during a 900 second simulation run with a pause time of zero for (a) uniformly distributed velocity of [1,19] ms^{-1}, and (b) uniformly distributed velocity of (0,20] ms^{-1}. Graph is acquired from Yoon, et al. (Yoon, Liu & Noble 2003).

For the case of [1,19] ms^{-1}, the average nodal speed of the entire network reaches steady-state (within 10% of the final average steady-state value) within 142 seconds. However, for the case of (0,20] ms^{-1}, the average nodal speed of the entire network continuously decreases over time.

The improvements recommended by Yoon, et al. (Yoon, Liu & Noble 2003) have been adopted in this thesis. In addition, the maximum period a node may maintain the same random waypoint is limited to t_{limit} seconds, as shown in equation 7.1, where the coordinates of the initial and destination point are (x_i, y_i) and (x_d, y_d), respectively, and v is the velocity.

$$t_{limit} \geq \frac{\sqrt{|x_i - x_d|^2 + |y_i - y_d|^2}}{v} \qquad \text{(Eq. 7.1)}$$

With these modifications, those slow moving nodes will be given the opportunity to choose a different random waypoint and a different velocity, while keeping the condition of $v_{\min} > 0$. The value of $t_{limit} = 30$ is adopted for the simulations on the basis that this will allow each node at least 30 waypoint changes during a simulation run of 900 seconds. Furthermore, an initial period of 200 seconds is allowed to elapse before any traffic sources are activated in order to ensure that the simulation is in steady-state in terms of mobility (Yoon, Liu & Noble 2003). The 200 seconds of time also provides the routing protocol more than sufficient time for each node to become fully aware of all the neighbouring nodes within its awareness region, as discussed later in Section 7.3.5.

7.2.3 Node Movement and Traffic Pattern Scenario Files

The node movement scenario files are generated using the Setdest program included with the NS-2 simulation environment software (NS-2 2005). The '*2003 U.Michigan*' version of the program was used to generate node movement, as revised by Yoon, et al. (Yoon, Liu & Noble 2003), to accommodate the author's modifications discussed previously. In total, 15 scenario files are generated for the 50-node random waypoint network topology with a uniformly distributed nodal velocity between 1 and 19 ms^{-1}, and a pause time of zero, as given in Table 7.1. This velocity profile is consistent with speeds of slow moving nodes such as walking at 3.6 km/h, as well as vehicular nodes up to 68.4 km/h. Each scenario consists of 900 seconds of simulated node movement which provides time for a substantial number of link and route changes to occur. Furthermore, a number of nodes will move away from their neighbouring nodes, and thus become unreachable for some periods of time during the 900 seconds. The mobility statistics for the generated node movement files is given in Table 7.2.

Scenario File	Link changes	Route changes	Destination Unreachables
1	13097	74845	292
2	13908	76063	335
3	13525	79627	145
4	13555	72989	294
5	13587	71842	458
6	13055	75489	288
7	12641	72410	429
8	13523	83141	141
9	12320	63209	456
10	12398	68564	239
11	14401	73195	233
12	11397	66575	519
13	12892	67133	147
14	14220	77613	423
15	12915	74556	147
Average	**13162**	**73150**	**303**

Table 7.2 – The generated node movement scenario files indicating the number of link and route changes, and the number of times a possible destination node can become unreachable during 900 seconds of simulation time.

Two different traffic sources are considered in the performance metric assessments; namely a user datagram protocol (UDP) traffic source, and a transmission control protocol (TCP) traffic source. The UDP source uses a constant-bit-rate (CBR) connectionless traffic generator to create packets at a constant bit rate at a specified packet size. The TCP source uses a file-transfer-protocol (FTP) connection-based traffic generator to create an infinite number of packets at a specified packet size for a given duration. Of interest is the unique *tcp_window* size parameter that can be set in order to change the packet window size of the TCP traffic source. This value corresponds to the number of TCP packets that are allowed to traverse the network at any point in time before being acknowledged with TCP acknowledgement packets.

Using the two different traffic sources, various scenarios can be created by varying the number of flows between source and destination nodes, and to demonstrate the effect on the underlying MAC protocol. For example, a CBR source is primarily

used to challenge the routing protocol to find optimal routing paths for outgoing data packets. In this case, maximising the traffic volume is not as important, as route changes might occur infrequently such that successive packets will follow the same routing path. For that reason, maximising throughput is more of a test for MAC protocol performance than routing protocol performance. On the other hand, a TCP source with various *tcp_window* size parameters can illustrate scenarios where the proposed routing protocol handles the traffic well, whereas other routing protocols can perform poorly due to the underlying MAC protocol.

7.3 ROUTING PROTOCOL PERFORMANCE

To compare the proposed MultiWARP routing protocol with the results of other research, different quantitative performance metrics must be assessed. In this section, the results of the subsequent performance metrics are discussed, as follows:

- Packet delivery ratio & throughput,
- End-to-end packet delay,
- Packet loss,
- Route discovery & acquisition delay,
- Routing overhead,
- Impact of routing reactivity on the awareness region,
- Route selection performance, and
- Routing stability.

Each performance metric assessment is made using a simulation run consisting of 15 node movement and traffic pattern scenario files, as discussed in Section 7.2.3. This allows for direct comparisons between results obtained for MultiWARP and DSR for any individual scenario file during the 900 seconds of simulation time. The data points from the 15 simulation runs are then used to calculate the average results for the MultiWARP and DSR routing protocols for each performance metric.

As discussed in Sections 4.3.2 and 4.6, the DSR routing protocol is chosen for simulation comparison with MultiWARP in this thesis. In research by Maltz, et al. (Maltz et al. 1999), it was determined that DSR is a sound candidate for demonstrating reactive routing protocols, and simulation results obtained from DSR generalise other reactive routing protocols using similar techniques. Therefore, the simulation results obtained using the DSR protocol is used to compare the performance of the reactive routing component of MultiWARP. Furthermore, in studies by Ehsan, et al. (Ehsan & Uzmi 2004), it was shown that DSR is the more suitable reactive routing protocol for ad-hoc networks, and outperforms AODV in a number of different scenarios with various metrics such as mobility and network size.

7.3.1 Packet Delivery Ratio

The influence of network topology changes on the ability of the MultiWARP routing protocol to deliver data packets to destination nodes is investigated in this section. The topology chosen is the 50-node random waypoint network with parameters according to Table 7.1. In this study, a constant-bit-rate (CBR) traffic load of 32 packets per second, divided into 8 flows of 4 packets per second, is offered to the network. Each of the 8 flows has a unique source and destination node, and the traffic sources are activated at 200 seconds and allowed to run until the end of the simulation time. The packets contain 64 bytes of data, and this suggests that the maximum system throughput that can be attained with this CBR traffic source is 16 kbps (16,384 bps). This capacity is well below the maximum transmission rate of 2 Mbps as specified by IEEE 802.11b physical layer to illustrate the effects of topological changes only.

The simulation illustrates how well the MultiWARP routing protocol behaves in an environment of random node movement in terms of successful packet delivery ratio and achieving smooth throughput. The packet delivery ratio is the ratio between the number of data packets generated by the application layer at the source node, and the

number of data packets received by the application layer at the destination node. This ratio measures the ability of the routing protocol to deliver the offered traffic load to their destination nodes by way of the intermediate nodes. Furthermore, it is a measure of how well the routing protocol performs in terms of its ability to correctly route data packets using a valid routing path, given that the destination node is indeed reachable.

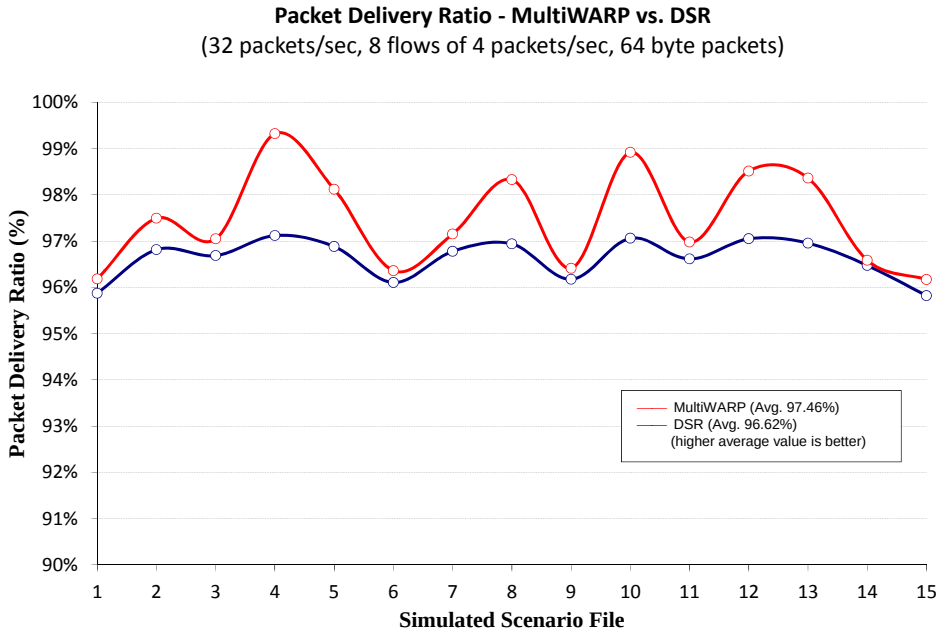


Figure 7.5 – Packet delivery ratio comparing the MultiWARP and DSR routing protocol for a traffic load offered at 32 packets per second.

As shown in Figure 7.5, MultiWARP outperforms DSR in all of the 15 scenarios, yielding an average packet delivery ratio of 97.46% compared to DSR's average packet delivery ratio of 96.62%, an improvement of 0.84%. Firstly, it should be noted that a packet delivery ratio of 100% packet is not achievable as during certain periods within the 900 seconds of simulation, some destination nodes became unreachable due to being physically out of transmission range. Secondly, the variation in the graph is not of interest, instead it shows MultiWARP has outperformed DSR in terms of absolute value in each scenario. The fluctuation

between the MultiWARP and DSR packet delivery ratio is due to the route selection criterion active in MultiWARP.

In MultiWARP, the optimum routing path is chosen at any given time for each data packet in order to reduce the routing path hop count, and in turn reduce end-to-end packet delay. With DSR, the routing path remains the same until the route is determined to be invalid. As a result, DSR could use a non-optimal routing path for a data packet at a particular time, yet still be able to deliver the packet albeit with a higher end-to-end packet delay. A comparison between the MultiWARP and DSR end-to-end packet delay is assessed in the next section.

The overall network throughput in kilobits per second (kbps) can be calculated by multiplying the packet delivery ratio, the offered traffic load, and the packet size in bits, as shown in equation 7.2.

$$T_{MultiWARP} = 0.9746 \times 32 \times (64 \times 8) = 15967.85 \text{ bps} \qquad T_{DSR} = 0.9662 \times 32 \times (64 \times 8) = 15830.22 \text{ bps} \qquad \text{(Eq. 7.2)}$$

The offered CBR traffic load was set to 16 kbps, thus using the simulation results the average network throughput for MultiWARP and DSR is 15.97 kbps and 15.83 kbps, respectively. An analysis of network throughput using a TCP traffic load is provided in Section 7.3.7.

7.3.2 End-to-End Packet Delay

The average network end-to-end packet delay (also termed packet latency) is the time taken for a packet to originate at the source node, traverse the network via the intermediate relay nodes, and arrive at the destination node. In this study, the packet delay is measured from end-to-end (i.e., not the round-trip delay) between the time of generation at the source node and successful arrival at the destination node. The packet delay takes into account the time taken for channel transmission and

propagation, MAC layer scheduling and retransmission, and routing layer processing at all the intermediate nodes as specified by the routing path. The average network end-to-end packet delay comparison between the MultiWARP and DSR routing protocols is shown in Figure 7.6.

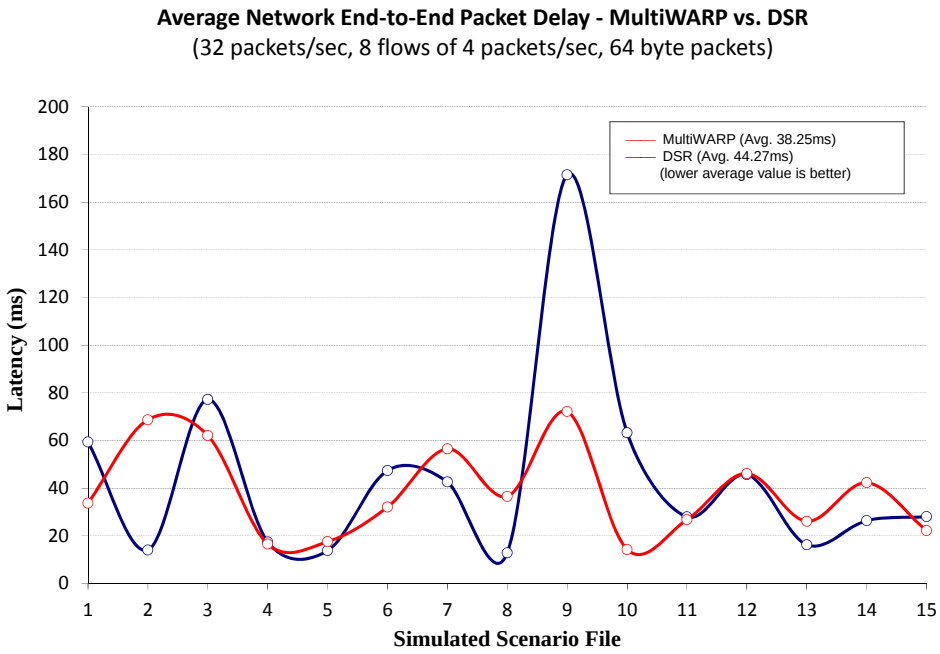


Figure 7.6 – Average network end-to-end packet delays of the MultiWARP and DSR routing protocols. Traffic load offered is 32 packets a second, divided into 8 unique source-destination flows of four 64-byte packets a second.

The MultiWARP routing protocol outperforms DSR in all but 6 of the 15 scenarios, yielding an average network end-to-end packet delay of 38.25 ms, compared to DSR's average network end-to-end packet delay of 44.27 ms. Overall, the graph indicates that MultiWARP has outperformed DSR by an average of 15.7% or 6.02 ms. In particular, simulated scenario #9 shows a marked decrease in the packet delay when using MultiWARP compared to DSR, namely 72.17 ms versus 171.51 ms, respectively. From inspection of the simulation trace file, the reason was due to DSR choosing a sub-optimal routing path which frequently resulted in routing failures and hence incurring packet delay. Next, the average network packet loss for the 15 scenarios is evaluated.

7.3.3 Packet Loss

In this section, the average network packet loss shows the amount of data packets that are discarded attributable to an error occurring in the lower layers; namely the physical or MAC layer. This can be due to either a transmission error caused by a collision, a receiving node being flagged as busy (MAC layer error), or due to an intermediate node being out of range of the transmitting node (physical layer error). As a consequence, the data packet is unable to traverse the network at that instant and can be considered as a loss in instantaneous throughput. Furthermore, it is possible for a data packet to be lost and then retransmitted multiple times at different intermediate relay nodes along the routing path as it traverses the network, and this will accumulate towards the packet loss total. It should be noted that this performance metric should not be confused with the packet drop ratio, which is equal to 1 subtract the packet delivery ratio, as defined in Section 7.3.1.

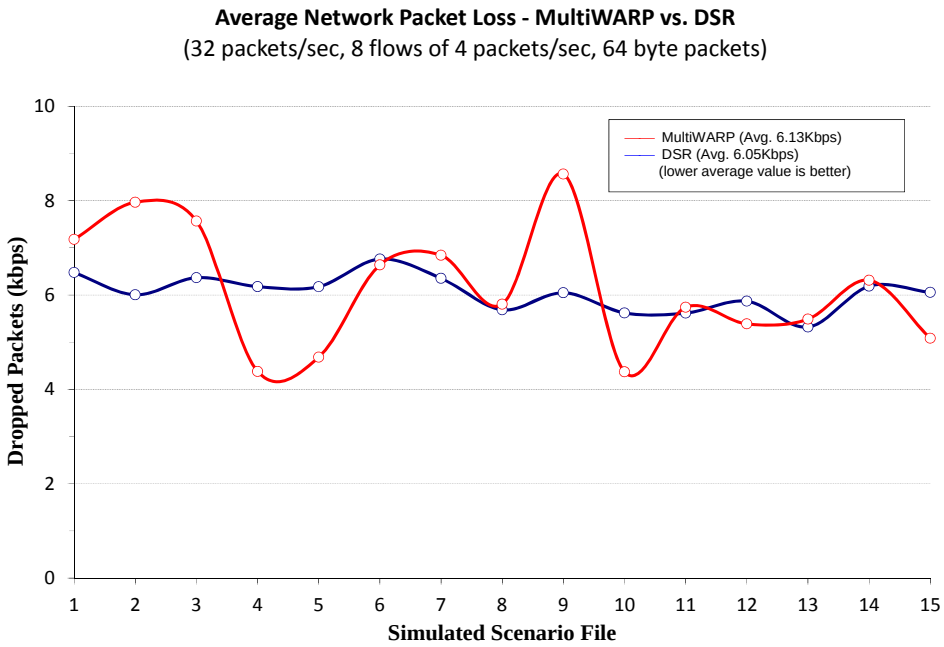


Figure 7.7 – Average network packet loss comparing the MultiWARP and DSR routing protocols for a traffic load offered at 32 packets a second, divided into 8 unique source-destination flows of four 64-byte packets a second.

As the IEEE 802.11 MAC layer is capable of retransmission, as discussed in Section 2.5, each subsequent transmission that fails will accumulate towards the packet loss total. The retransmission process continues until the current node has successfully transmitted its data packet, or has exceeded the number of allowed retransmissions before the data packet is ultimately dropped, as discussed in Section 2.5.1. The average network packet loss, as shown in Figure 7.7, represents all packet losses relating to the CBR traffic load offered to the network (including retransmissions) for both the MultiWARP and DSR routing protocols.

From inspection of the simulation trace files, the resultant packet loss is primarily due to packets exceeding the retransmission limits imposed by the MAC layer. The total network packet loss can be divided into two groups; those packets which were lost but were able to be successfully retransmitted some time later, and those which were unable to be successfully retransmitted. The percentage of packets that upon retransmission were successfully received at the destination node was 12.06% for MultiWARP and 7.13% for DSR. However, for the majority of packets, 87.94% for MultiWARP and 92.87% for DSR, retransmission attempts led to further collisions which resulted in the packet being ultimately dropped after exceeding the retransmission limit.

Overall, the difference in average network packet loss between MultiWARP and DSR is 0.08 kbps, and can be considered negligible. Furthermore, the packet loss and subsequent retransmission failures are due to high nodal mobility as observed from the simulation trace files. In particular, for scenario files #2, #3, and #9 MultiWARP chose the optimum route using the least hops criterion at a particular time instant. However, two of the intermediate relay nodes within the routing path were travelling at over $33ms^{-1}$ away from each other, on multiple occasions. This led to the intermediate relay nodes becoming out of range of each other, and causing the unfavourable situation of high MAC layer retransmission failure before a link-failure is reported to the routing layer, as discussed in Section 2.5.3.

In all cases, however, MultiWARP managed to deliver all the data packets involved to the destination node, and outperformed DSR in terms of the overall packet delivery ratio. In particular, for scenario file #9, DSR resulted in an overall end-to-

end packet delay of 171.51 ms compared to 72.17 ms for MultiWARP. This was because DSR chose a routing path that consisted of intermediate relay nodes with a lower node mobility; moving at a maximum of $6.5ms^{-1}$ away from each other during the same time period when MultiWARP experienced $33ms^{-1}$. This can be improved by using an alternative metric than hop-count alone, such as one that encompasses nodal velocity, but is outside the scope of this thesis. MultiWARP does support the inclusion of using other metric information such as congestion control information and bandwidth availability using the *Options* field, as discussed in Section 5.3.

7.3.4 Route Discovery & Acquisition Delay

The time required to discover a valid routing path in any routing protocol is crucial in a wireless ad hoc network, and plays an important role in determining the responsiveness of the network. Before a data packet can be transmitted, the routing protocol must first insert a valid source route into the packet header before passing it to the MAC layer for transmission. The advantage in using a hybrid routing protocol like MultiWARP lies in the fact that valid routing paths are virtually instantaneously available to the routing layer. This is the case if the destination node lies within the awareness region (apart from the processing delay which can be considered negligible). For nodes that lie outside of the awareness region, a reactive component that performs the route request (RREQ) process is initiated, and is discussed in Chapter 6.

In MultiWARP the RREQ process uses the covercasting technique, which is detailed in Sections 6.2 and 6.3, and does not rely on inefficient flooding. Consequently, the time that elapses before a valid routing path is returned via a route reply (RREP) packet is much reduced. To determine the route discovery times for the MultiWARP and DSR routing protocols, a simulation of an ad-hoc network consisting of 50 mobile nodes using the random waypoint topology is carried out. The time it takes to discover a valid routing path is monitored for all 15 different node movement scenario files for the duration of the simulation. Furthermore, each scenario consists of 8 unique source-destination traffic flows in order to trigger the route discovery

procedure within the routing protocol. The network scenario parameters are as given in Table 7.1, and the results obtained from simulation are shown in Table 7.3.

	Average	Minimum	Maximum	Std. Deviation
MultiWARP	16.07 ms	0.75 ms	199.86 ms	22.49 ms
DSR	540.43 ms	14.72 ms	10258.48 ms	844.47 ms

Table 7.3 – The average, minimum, maximum, and standard deviation of the time it takes to discover and acquire a valid routing path for MultiWARP and DSR using the RREQ process.

The resultant reduction in the average time it takes to discover and acquire a valid routing path comparing MultiWARP and DSR is considerable, 16.07 ms versus 540.43 ms, or alternatively, a 33.6 fold decrease in time before a data packet can be transmitted. Through use of the covercasting technique, the maximum time taken to discover a valid routing path was only 199.86 ms for MultiWARP, compared to over 10.2 seconds for DSR. The stability of the route discovery mechanism can be justified by examining the standard deviation, 22.49 ms for MultiWARP and 844.47 ms for DSR before a valid routing path is acquired for the 15 simulated scenarios. Furthermore, in DSR a stale route is removed from the route cache only when a RERR packet explicitly instructs it to do so, as routing paths remain valid indefinitely due to the reactive nature of the DSR protocol. In brief, the advantage in using a hybrid routing protocol results in a markedly decreased time it takes to discover a valid routing path before packet transmission can take place.

The removal of the reliance on the address resolution protocol (ARP) in MultiWARP has aided in the significant decrease in routing path acquisition. This is because the MAC address of the source node is carried in the RUPDT packet, at a cost of only 6 bytes per RUPDT packet every μ seconds, as discussed in Section 5.5. Therefore, ARP packets are no longer required to resolve MAC addresses and this has resulted in a significant time saving in the proactive case. In the next section, the routing overhead that is incurred by the proactive component of the proposed routing protocol is studied.

7.3.5 Routing Overhead & Impact of Routing Reactivity on Awareness Region

The size of the routing table maintained at each node is primarily governed by the size of the awareness region. In general, a larger routing table implies that an increased number of alternative routing paths are available. As discussed in Section 5.2.4, the awareness region parameter R influences the reactivity of the protocol. For example, a densely populated network should have a low value for R as many nodes are within transmission range of one another, causing excessively large routing tables and RUPDT packets. Conversely, a sparsely populated network should have a high value of R because the routing table and RUPDT packets will remain small. It should be noted that setting $R = 1$ will result in MultiWARP to behave purely as a reactive routing protocol, and hence it will have no RUPDT overhead what-so-ever. On the other hand, setting R to a very large value (or equal to a hop-count value larger than any routing path possible within the network) makes MultiWARP purely proactive. Neither option is attractive for a multi-hop wireless ad-hoc environment due to excessive route discovery latency or overhead concerns, respectively.

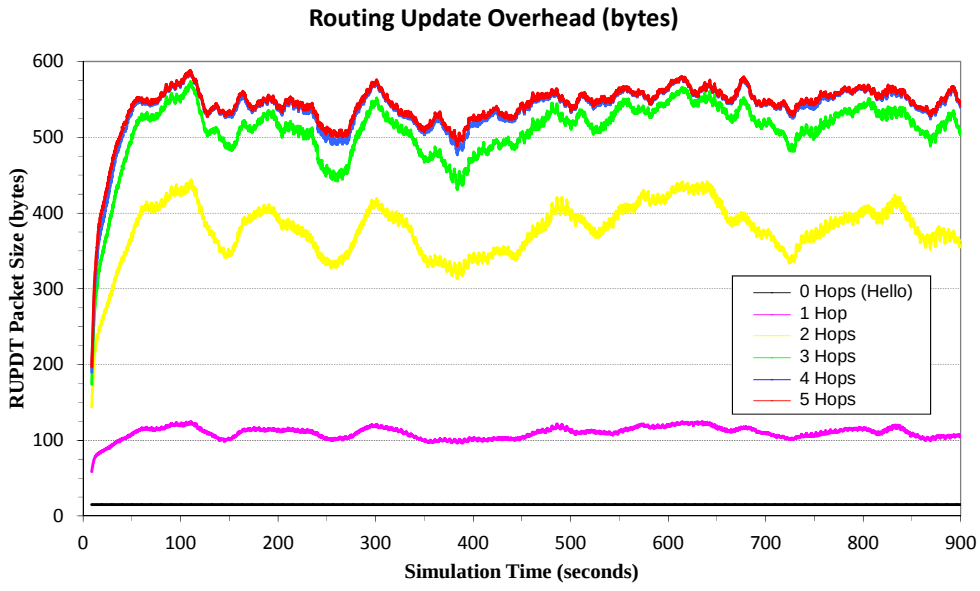


Figure 7.8 – Effect of varying the awareness region parameter R on the observed RUPDT packet size in bytes for a 50-node random waypoint network topology.

By changing the awareness region parameter R (to vary the size of the awareness region in terms of hop-count), it becomes possible to study the effect on the RUPDT packet size. Again, the topology chosen is the 50-node random waypoint network with parameters according to Table 7.1. However, in this study, no traffic load is offered to the network so as to allow only the routing overhead to be observed. The RUPDT packet size for each of the 50 nodes is then observed over the 900 seconds of simulation time for the 15 node movement scenarios. The data points from the 15 simulation runs are then used to calculate the average RUPDT packet size, and is repeated for $R = \{1, 2, 3, 4, 5\}$, as shown in Figure 7.8.

The graph in Figure 7.8 depicts the average instantaneous packet size of the RUPDT packet that is transmitted every μ seconds by each node, where $\mu = 5$ seconds. The average routing overhead in bytes per second per node is determined by dividing the average instantaneous packet size by μ. From the results, it is observed that the average routing overhead is approximately $500/\mu = 100$ bytes per second per node for an awareness region of $R = \{3, 4, 5\}$ hops. Similarly, the overhead for $R = \{1, 2\}$ is 20 and 70 bytes per second, respectively.

In Section 5.2.1, it was found that any new node added to an existing network is automatically self-configured within a maximum of μ seconds after start-up. Furthermore, it was found that neighbouring nodes within the awareness region will become aware of the new node within the minimum time of $t_{\min} = d_{prop}(R)$ seconds and a maximum time of $t_{\max} = \mu(R-1) + d_{prop}(R)$ seconds, where d_{prop} is the propagation delay between nodes. In this study, all 50 nodes are simultaneously switched on at time $t = 0$, causing an initial start-up phase whereby the nodes are progressively starting to receive routing information about one another.
Focusing on the time period $t = [0..50]$, as shown in Figure 7.9, it is recognised that the trend-line tends to become stable, rather than flat, after a maximum time of $t_{\max} = \mu(R-1) + d_{prop}(R)$ seconds. This is due to the effects of mobility, allowing nodes to come into and move out of another nodes awareness region, causing some stale nodes to accumulate as well as the discovery of these new nodes. This thus alters the size of the routing table and hence the RUPDT packet size. Therefore,

after a maximum of $t_{max} = \{5, 10, 15, 20, 25\}$ seconds for $R = \{1, 2, 3, 4, 5\}$, respectively, each node becomes aware of all the neighbouring nodes within its awareness region.

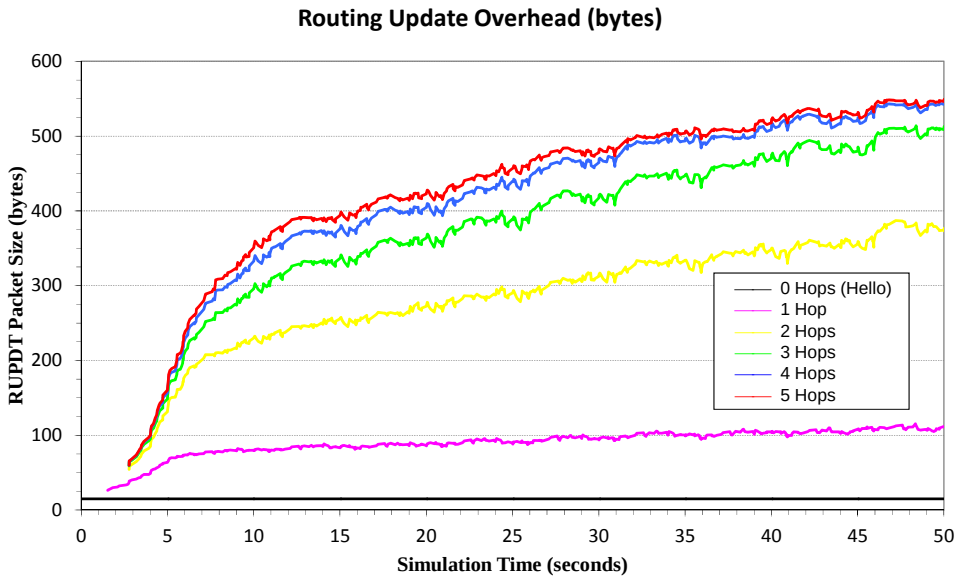


Figure 7.9 – This shows the same results as in Figure 7.8, but with emphasis on the time period $t = [0..50]$ showing the trend-line more clearly.

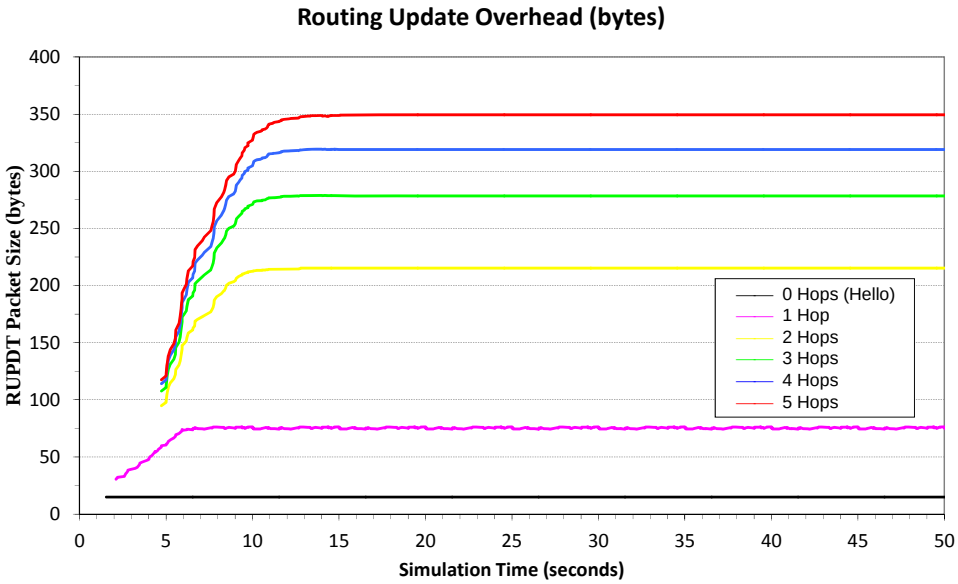


Figure 7.10 – Effect of varying the awareness region parameter R on the observed RUPDT packet size in bytes for a 50-node fixed network topology, with emphasis on the time period $t = [0..50]$.

If the effects of mobility are removed, the trend-line is expected to become flat after a maximum time of $t_{\max} = \mu(R-1) + d_{prop}(R)$ seconds. To confirm this, the 50-node random waypoint network topology is converted to a 50-node fixed network topology by setting the nodal velocity equal to $v = 0$. Again, no traffic load is offered to the network, and the RUPDT packet size for each node is observed for 900 seconds. The data points from the 15 simulation runs are then used to calculate the average RUPDT packet size, and is repeated for $R = \{1, 2, 3, 4, 5\}$, as shown in Figure 7.10.

Focusing on the time period $t = [0..50]$, as shown in Figure 7.10, it is identified that the maximum time condition holds for all $R = \{1, 2, 3, 4, 5\}$, and that the trend-line becomes flat before $t_{\max} = \{5, 10, 15, 20, 25\}$ seconds has elapsed, respectively. It is observed that increasing the awareness region parameter R results in a diminishing increase in the size in RUPDT packet. This is because the routing path to a node X hops away is reached via a path that is (X-1) hops away, which is already known within the routing table. Therefore, only the incremental change in routing information needs to be inserted into the RUPDT packet. It should be noted that the routing information becomes more stale the further the node lies away from the source node, as discussed in Section 5.2.3. This states that the stale node removal algorithm depends on $\mu(C)(R)$, where μ is the periodic update interval, C is an arbitrary timeout constant (normally $C = R$), and R is the awareness region in hops.

7.3.6 Route Selection Criteria

In this section, the effectiveness and performance of MultiWARP's two secondary route selection criteria as discussed in Section 6.6 are evaluated. Namely, the uplink integrity method and the at-random method. These methods provide a means for traffic-based load balancing by distributing traffic load among intermediate relay nodes (Toh, Le & Cho 2009). The primary route selection scheme is based upon the shortest hop-count of the routing path, but frequently results in many alternative routing paths. As a result, any of the alternative routing paths can be selected in

order to reach the destination node, and this selection is made by the secondary route selection criterion.

To determine the effectiveness of the two methods, a simulation of an ad-hoc network consisting of 50 mobile nodes using the random waypoint topology is carried out. The network usage in terms of throughput activity per node and traffic latency is monitored for the duration of the simulation. The network scenario parameters are as given in Table 7.1, with the addition of a TCP traffic source and a TCP sink between the source and destination nodes, respectively.

The TCP traffic source consists of 1500 byte data packets travelling from the source node to the destination node, with TCP acknowledgement packets travelling back towards the source node. It should be noted that TCP acknowledgement packets do not necessarily use the reversed source-route path back to the source node. The TCP traffic source has its *tcp_window* size parameter set to 1 in order to simulate a stop-and-go type protocol, i.e., the data packet must be acknowledged before another can be transmitted. This is done pre-emptively to ignore the effects of any contention problems that may arise within the medium access control (MAC) layer. The problems that can arise are due to multiple packets traversing the network, which in turn can cause the network allocation vector (NAV) to be set incorrectly, and is discussed later in Section 7.3.8.

The source and destination node pair is identified to be initially separated by 4 hops at time $t = 50$ seconds. Through random waypoint movement, the source and destination node pair move closer together until they become immediate neighbours (i.e., separated by 1 hop) at time $t = 160$ seconds for a duration of 40 seconds. The network usage in terms of throughput activity per node is then observed for the time period $t = [50..200]$. As confirmed in Section 7.3.5, the 50 nodes will have become fully aware of all the neighbouring nodes within the awareness region before $t = 50$ seconds has elapsed. Furthermore, this simulation is not concerned about steady-state mobility of all 50 nodes, as a specific source and destination node pair is identified and tracked. Thus waiting a period of 200 seconds to elapse is not

necessary in this simulation. The results obtained for the secondary route selection criterion based on the uplink integrity method is shown in Figure 7.11. On the other hand, when the at-random method is chosen to be the secondary route selection criterion, the results obtained are shown in Figure 7.12.

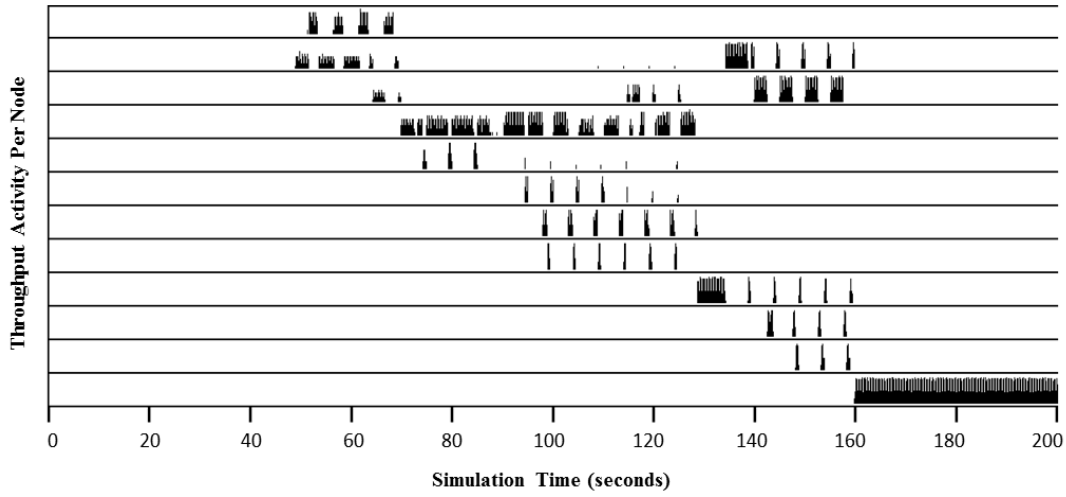


Figure 7.11 – The distribution of the TCP data packet flow between various intermediate nodes when the uplink integrity method is adopted for the secondary route selection criterion.

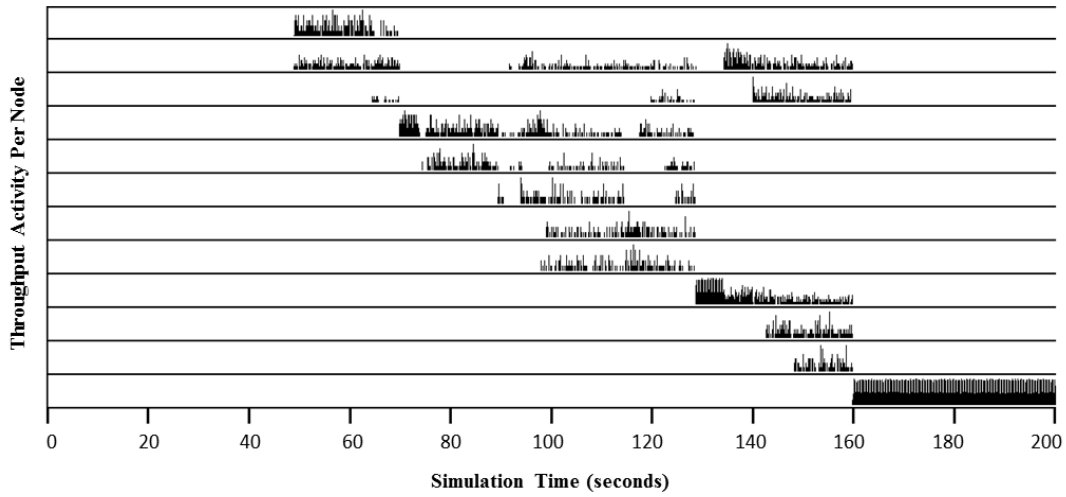


Figure 7.12 – The distribution of the TCP data packet flow between various intermediate nodes when the at-random method is adopted for the secondary route selection criterion.

In both Figure 7.11 and Figure 7.12, the horizontally stacked rows represent the throughput activity per node as they become utilised during the time period $t = [50..200]$. In other words, these nodes represent intermediate relay nodes forwarding the TCP packets from the source node towards the destination node. It is observed that three nodes are utilised between $t = [50..75]$ seconds, and two nodes are utilised between $t = [75..90]$ seconds. Hereafter, an increase in the number of possible routing paths (due to node mobility) allows for more intermediate relay nodes to carry the traffic load. As such, between $t = [90..120]$ seconds the number of nodes utilised is increased to six, and finally seven nodes between $t = [120..130]$ seconds. After this time, between $t = [130..160]$ seconds, the number of intermediate relay nodes carrying traffic is reduced to five. Upon inspection of the simulation trace file, at $t = 130$ seconds the hop-count distance between the source and destination node is decreased to 2 hops. This caused five nodes belonging to a group of longer 3 hop routing paths to be eliminated, whilst 3 additional nodes were now being favoured using the shorter 2 hop routing path. Due to the source and destination nodes moving within closer proximity to each other, between $t = [160..200]$ seconds the nodes become immediate neighbours and therefore communicate directly using only a single-hop.

7.3.7 TCP Throughput & TCP Latency

The difference between the two alternative selection criteria, discussed in the previous section, is in the diversity of network usage or distribution of traffic load. The uplink integrity method tending to have traffic focused on one node and then switching to another node in bursts, and the at-random method spreading the traffic out over multiple nodes simultaneously. Overall, the summation of the TCP packets received at the destination node is the same for both secondary selection criteria. The throughput and latency is recorded for the duration of $t = [50..200]$ seconds, as shown in Figure 7.13 and Figure 7.14, respectively, and was determined to be the same for both methods.

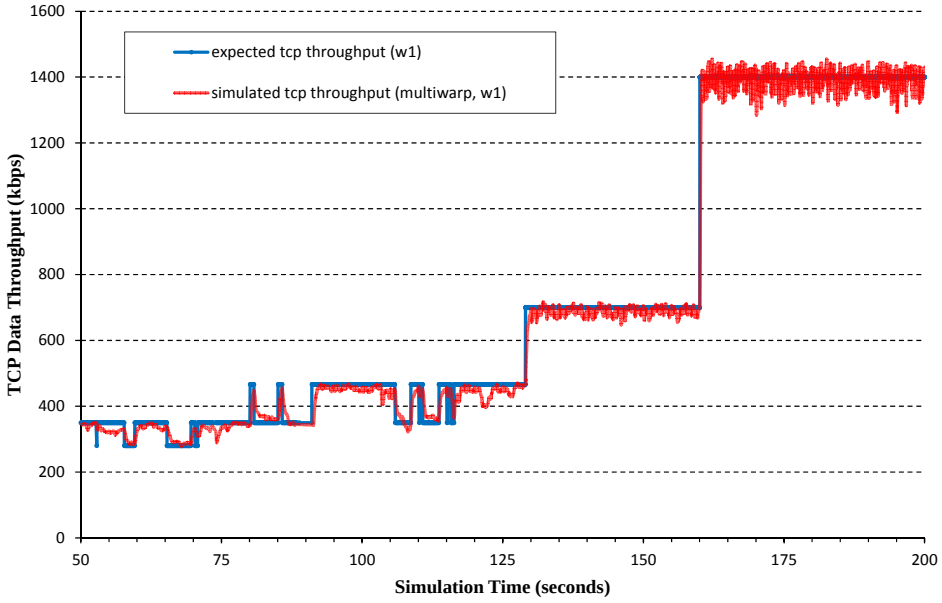


Figure 7.13 – Average TCP data throughput with *tcp_window* size parameter equal to 1, shown in red. The blue curve corresponds to the expected throughput divided by the hop-count distance between the source and destination nodes during simulation.

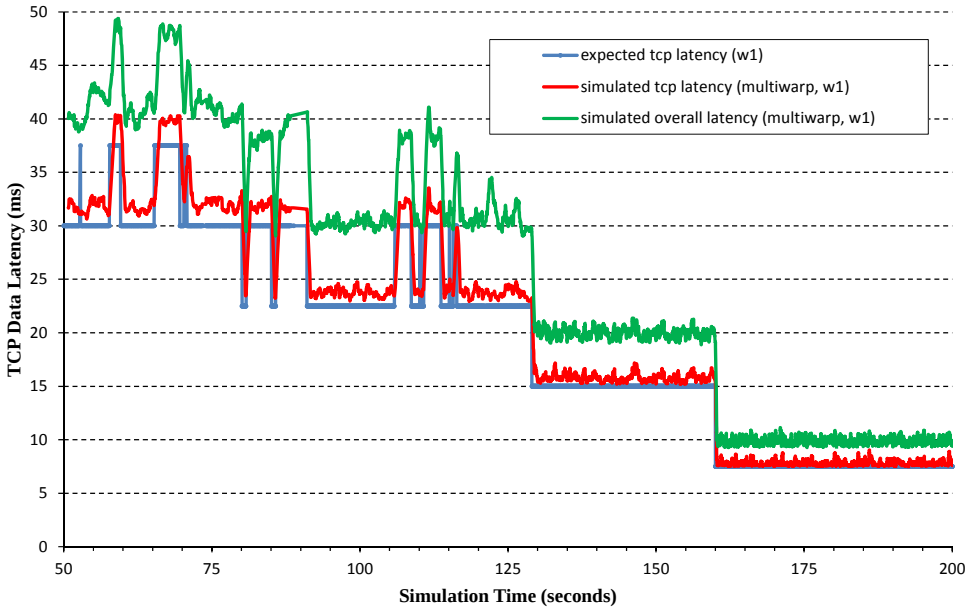


Figure 7.14 – Average TCP data latency with *tcp_window* size parameter equal to 1 for TCP data packets (shown in red), and the overall TCP transmission (shown in green). The blue curve corresponds to the expected latency multiplied by the hop-count distance between the source and destination nodes during simulation.

In Figure 7.13, the blue curve depicts the expected TCP data throughput attainable during the simulation. The expected throughput is dependent on the instantaneous minimum hop-count between the source and destination nodes, as observed from the simulation trace file. As such, a decrease in the number of hops results in a proportional increase in the throughput. That is to say, for the time periods $t = [50..90]$, $t = [90..130]$, $t = [130..160]$, and $t = [160..200]$, the minimum hop-count between the source and destination nodes are equal to 4, 3, 2, and 1, respectively. This equates to the throughput observed by the destination node (as shown in red) as fractions of the maximum single-hop throughput available divided by the hop-count; namely, 350 kbps, 466 kbps, 700 kbps, and 1.4 Mbps, respectively.

In Figure 7.14, the expected latency shown by the blue curve is observed to be proportional to the instantaneous minimum hop-count between the source and destination nodes. In other words, the latency experienced by a TCP data packet is equal to the latency for a single-hop communication multiplied by the instantaneous minimum hop-count between the source and destination node. The TCP data packet latency (shown in red) is closely matched to the expected latency for the length of the simulation duration. This latency describes the time it takes for a packet to be transmitted by the source node and received at the destination node only; thus the single-hop communication latency is determined to be 7.5 ms. This value includes the MAC layer data transmission handshake (RTS-CTS-DATA-ACK), but does not include the duration of the subsequent TCP acknowledgement packet. The overall latency of the TCP transmission, including the duration of the TCP acknowledgement packet and its accompanying MAC layer data transmission handshake, is shown by the green curve. The overall latency between subsequent TCP data packet arrival using a *tcp_window* size parameter of 1 is therefore determined to be 10.0 ms.

The expected TCP throughput and latency curves, as shown in Figure 7.13 and Figure 7.14, respectively, are used as a baseline for further comparisons. In the next section, the *tcp_window* size parameter is increased from 1 to 4 and 8 in order to compare the performance and behaviour of MultiWARP and DSR. In particular, the

effect of the TCP instability problem which occurs when the *tcp_window* size parameter is increased can be studied.

7.3.8 Routing Stability and Impact on TCP Throughput & TCP Latency

The presence of TCP instability in the IEEE 802.11 MAC layer is the result of the exposed node problem and this can lead to contention problems (Xu & Saadawi 2001), as discussed in Section 2.5.2. This problem can be noticed when large values of the *tcp_window* size parameter are used for the TCP traffic source, and the large number of retransmissions that are allowed by the MAC layer (Li, Kong & Chua 2007). The instability is caused by the MAC layer being unable to contact the subsequent intermediate relay node due to the network allocation vector (NAV) being set to "busy" at that node. This will directly cause the request-to-send (RTS) retransmissions to fail, causing a MAC layer link-failure to be reported to the routing layer.

Routing protocols such as DSR will drop all packets in the queue destined for the offending node after a link-failure has occurred and transmit a RERR packet back to the source node (Xu & Saadawi 2001). This leads to DSR having to redeploy its route discovery procedure to find a new routing path, during which the TCP transmission can timeout causing throughput loss. If a new routing path is found after some time, the TCP session will resume using a slow-start mechanism effectively reducing maximum throughput during this period. In contrast, MultiWARP will set the offending node to the inactive "expired" state, as discussed in Section 6.2.3, and will not be able to use the offending node until it has been reactivated. As MultiWARP can utilise multiple alternative routing paths that are already available in the routing table, it remains able to transmit data to the destination node by bypassing the offending node. Furthermore, it is able to do so without having to deploy a route discovery procedure. In this way, MultiWARP is able to effectively circumvent the TCP instability problem present in the IEEE 802.11 MAC layer.

To illustrate this point, a similar simulation scenario is created as in Section 7.3.6, but using an increased *tcp_window* size parameter of 4 and 8 instead of the original value of 1. It is expected that increasing the *tcp_window* size parameter will result in a higher throughput as multiple TCP data packets are able to traverse the network. The resultant throughput obtained using the MultiWARP (up-link integrity method as the secondary route selection criterion) and DSR routing protocol is shown in Figure 7.15(a–d).

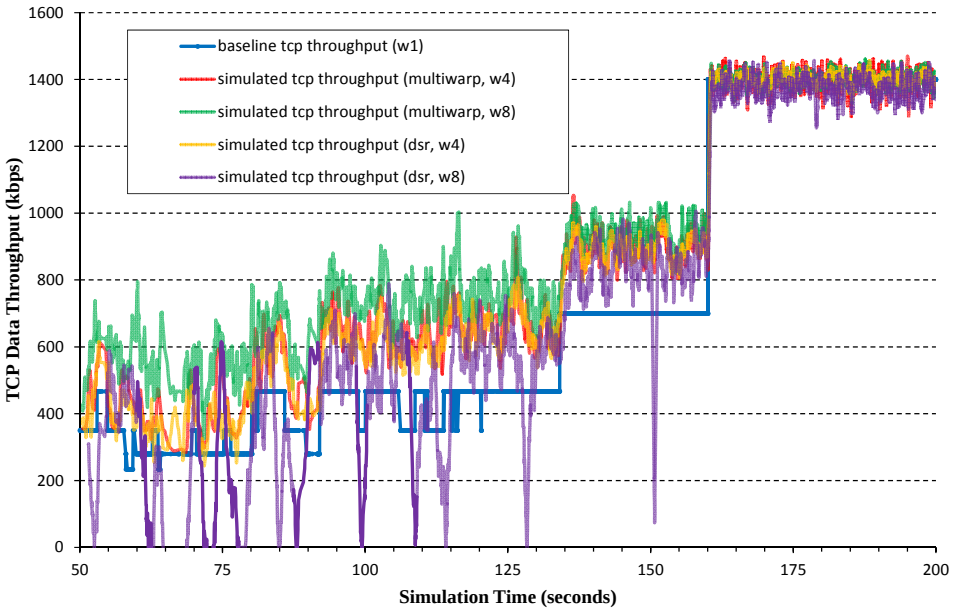
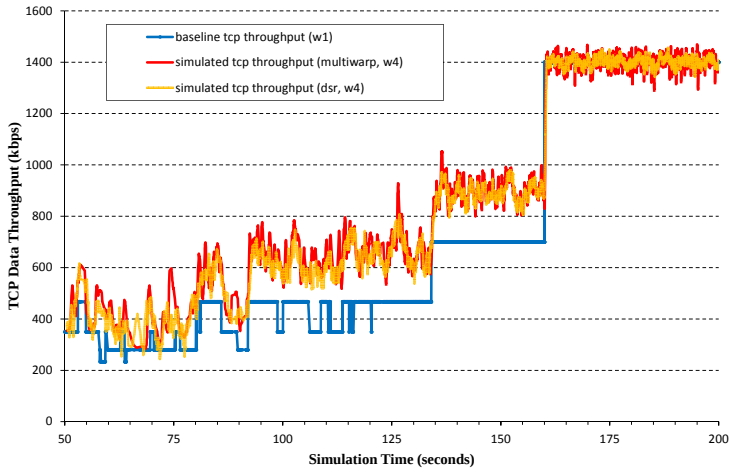
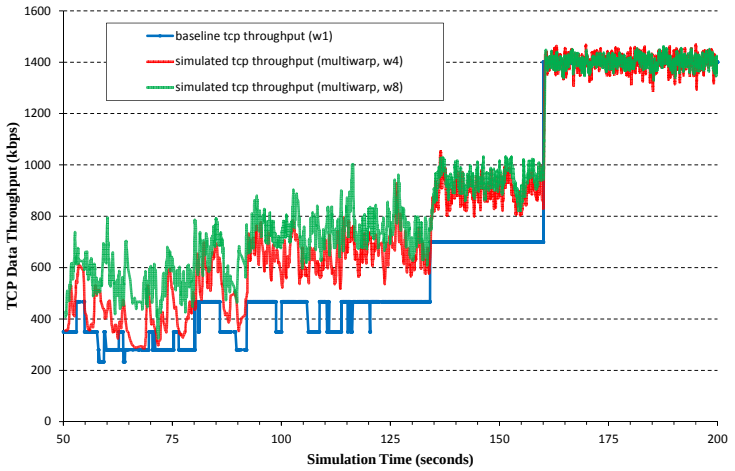


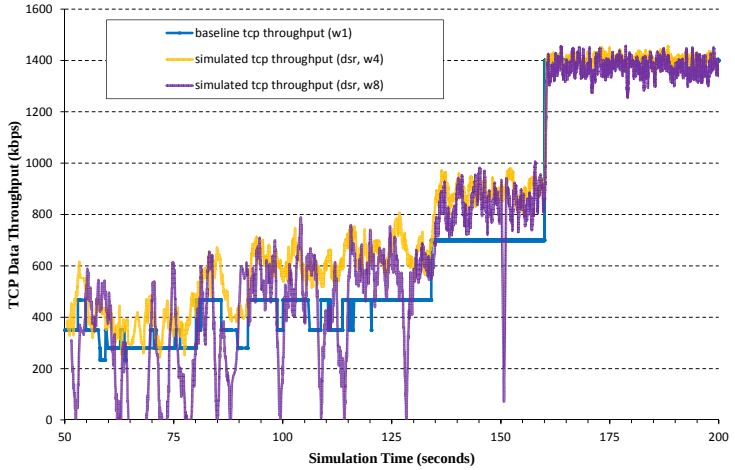
Figure 7.15 – Average TCP data throughput with *tcp_window* size parameter equal to 4 for MultiWARP is shown in red and DSR is shown in orange. For *tcp_window* size parameter equal to 8, MultiWARP is shown in green and DSR is shown in purple. The blue curve corresponds to the baseline throughput (for *tcp_window* size parameter of 1) divided by the hop-count distance between the source and destination nodes during simulation. This graph is depicted in further detail overleaf, showing the 4 throughput components in comparison to each other.



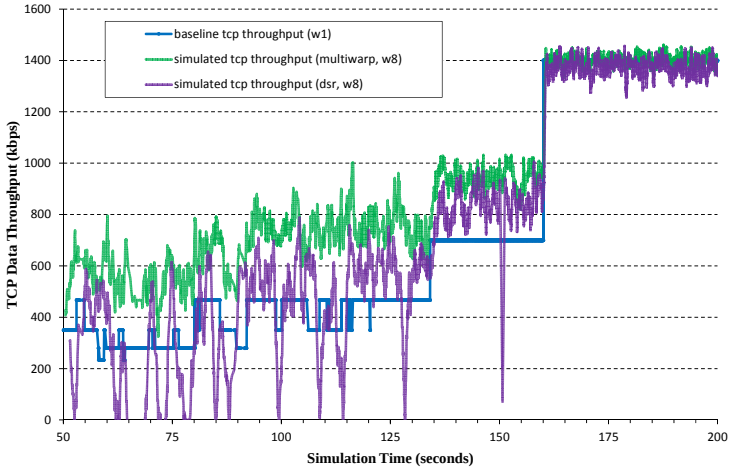
(a) MultiWARP *tcp_window* 4 , DSR *tcp_window* 4



(b) MultiWARP *tcp_window* 4 , MultiWARP *tcp_window* 8



(c) DSR *tcp_window* 4 , DSR *tcp_window* 8



(d) MultiWARP *tcp_window* 8 , DSR *tcp_window* 8

Figure 7.15(a–d) – Average TCP data throughput with *tcp_window* size parameter equal to 4 for MultiWARP and DSR, as indicated.

In Figure 7.15(a), it is observed that increasing the *tcp_window* size parameter from 1 to 4 results in an overall increase in throughput at the destination node. Comparing MultiWARP and DSR performance, MultiWARP achieves greater throughput during the time period $t = [50..135]$ and similar performance during the time period $t = [135..200]$. When compared to the baseline case with a *tcp_window* size parameter of 1, as shown by the blue curve, a higher throughput is noticed during the time periods $t = [50..90]$, $t = [90..135]$, and $t = [135..160]$. These time periods correspond to the source and destination nodes being separated by a hop-count of 4, 3, and 2, respectively. The average throughput during these time periods is 435 kbps, 640 kbps, and 900 kbps, an increase of 85 kbps, 174 kbps, and 200 kbps when compared to Figure 7.13, respectively. The throughput when the source node and destination node become immediate neighbours (i.e., a hop-count of 1) remains unchanged at 1.4 Mbps.

When the *tcp_window* size parameter is increased from 4 to 8, it is expected that additional throughput can be achieved. However, the *tcp_window* size parameter cannot be increased indefinitely in order to obtain additional throughput. This is due to the fact that there exists a maximum number of TCP packets that can traverse the network at any one time or be buffered within the packet queues at the intermediate relay nodes. These buffered packets increase the contention of the channel and leads to an overall reduction in throughput, a condition also noted by Fu, et al. (Fu et al. 2003). The optimal *tcp_window* length used by a TCP session stabilises at approximately 8 in this simulation scenario, such that throughput gains are not attainable if the *tcp_window* size parameter is increased to 16 or 32. In research by Fu, et al. (Fu et al. 2003), it was found that increasing the *tcp_window* size parameter beyond the optimal value can lead to a significant reduction in TCP throughput of up to 21% in random waypoint topologies.

In Figure 7.15(b), increasing the *tcp_window* size parameter from 4 to 8 has resulted in an increase in overall throughput using MultiWARP, especially during the time period $t = [50..160]$. The opposite effect is encountered for DSR, such that severe degradation in TCP performance is observed for the duration of the simulation, and

is shown in Figure 7.15(c). The throughput reaches zero on 12 occasions during this time, which coincides when the source and destination nodes are separated by more than 1 hop. This is caused by the MAC layer being unable to contact the next intermediate relay node due to the NAV being set to "busy". This is attributable due to two different situations. Firstly, multiple TCP data packets are being transmitted by nearby intermediate relay nodes (up to a maximum of *tcp_window* packets) before a TCP acknowledgement packet must be returned. Depending on which intermediate relay node has successfully contended for access to the channel, this node will transmit its TCP data packet first. Therefore, the situation can arise whereby two (or more) consecutive transmissions are carried out by the same intermediate relay node. Secondly, this prevents the TCP acknowledgement packets from being returned back to the source node leading to the TCP session to timeout. After the throughput has reached zero and a new route has been acquired by DSR, the TCP session restarts using a slow-start mechanism further reducing overall throughput.

MultiWARP outperforms DSR when the *tcp_window* size parameter is set to 8 as it does not suffer from this problem, as shown in Figure 7.15(d). The explanation why MultiWARP outperforms DSR is due to having multiple alternative routing paths available. The MultiWARP routing protocol allows the source node (or intermediate relay nodes) to adaptively distribute its TCP traffic load among all the alternative routing paths available. Furthermore, if the alterative paths are disjoint paths, i.e., they have no intermediate relay nodes in common, then TCP data packets are able to traverse the network with less bottleneck congestion. It should be noted, however, that in non-ideal scenarios the disjoint paths are not necessarily physically separated in terms of transmission or interference range.

Before analysing the TCP data packet latency when the *tcp_window* size parameter is increased above 1, it should be noted that primarily two factors affect latency performance. Firstly, TCP packets (both data and acknowledgement) do not need to traverse the network in sequential order (herein referred to as "ordered"). In other words, the sequence number of the TCP packet does not dictate the order of arrival at the destination node. If say 2 packets are transmitted by a source node in succession, it does not imply the destination node will receive those same 2 packets in succession, or even in that order (herein referred to as "unordered"). This is due to different

channel contention conditions between the intermediate relay nodes, allowing packets held in the packet buffers to be transmitted before or after other packets traversing the network. An inspection of how many consecutive TCP data and TCP acknowledgement packets are transmitted and received by the source and destination nodes in succession is given in Table 7.4 for a *tcp_window* size parameter of 4 using MultiWARP.

	1 Consecutive	2 Consecutive	3 Consecutive	4 Consecutive
TCP Data Sent	59.4%	28.5%	9.4%	2.7%
TCP Data Recv	58.9%	30.0%	8.6%	2.5%
TCP ACK Sent	58.6%	30.0%	8.8%	2.5%
TCP ACK Recv	59.8%	28.6%	9.2%	2.4%

Table 7.4 – Percentage of how many consecutive TCP data and acknowledgement packets are transmitted and received by the source and destination nodes with the *tcp_window* size parameter equal to 4 for the MultiWARP routing protocol.

It is noticed from Table 7.4 that approximately 90% of TCP data packets arrive at the destination node in either a succession of 1 or 2 consecutive packets. Similarly, the TCP acknowledgement packet is transmitted back towards the source node in much the same way. This affects the delay profile experienced at the destination node in terms of TCP data latency, as will be discussed later.

The second factor affecting TCP data packet latency performance is because of packet collisions due to the increased traffic load. For the case when the *tcp_window* size parameter is set to 4, there are 2417 request to send (RTS) packets that caused a collision out of a total 32,959 RTS events during the simulation. Most of these collisions occurred with other RTS packets, but more importantly, also caused collisions with 87 TCP data packets and 18 TCP acknowledgement packets. Furthermore, 174 RTS packets and 48 TCP data packets exceeded subsequent retransmission limits, as discussed in Section 2.5.1, causing them to be dropped.

As a result of multiple TCP data packets traversing the network, it is expected that the overall TCP data packet latency should be decreased. In Figure 7.16(a), the red dashed curve shows in fact that the ordered TCP data packet latency has increased when compared to Figure 7.14. In this case, the latency has increased because packets are arriving out of sequence order, thus the packets will incur delays within the packet buffer at the destination node after having physically arrived. The destination node will then proceed to wait for the packets with lower sequence numbers to arrive before passing the TCP data packets up to a higher layer.

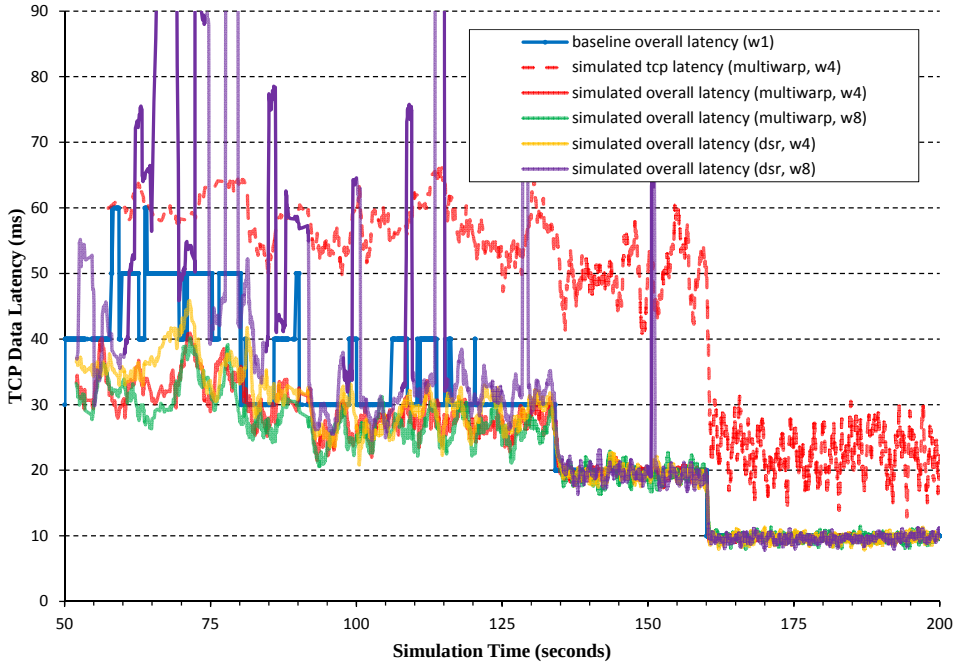
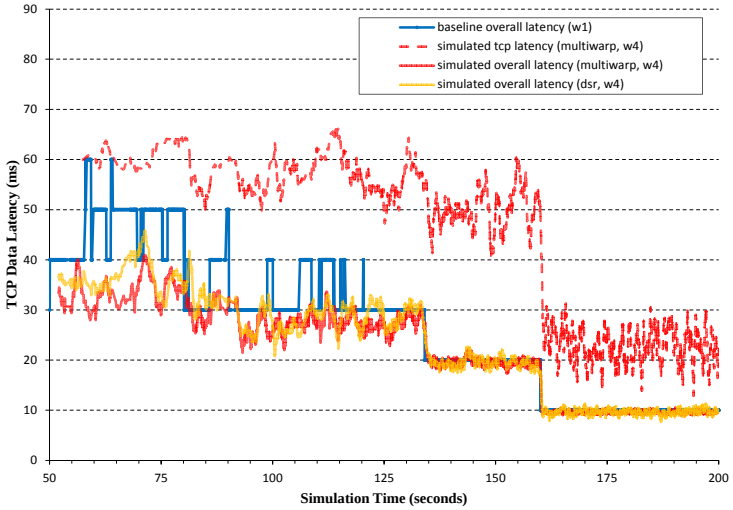
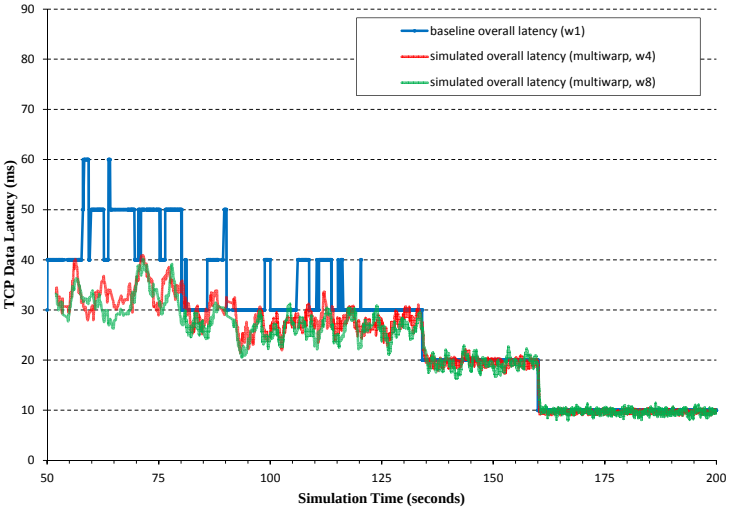


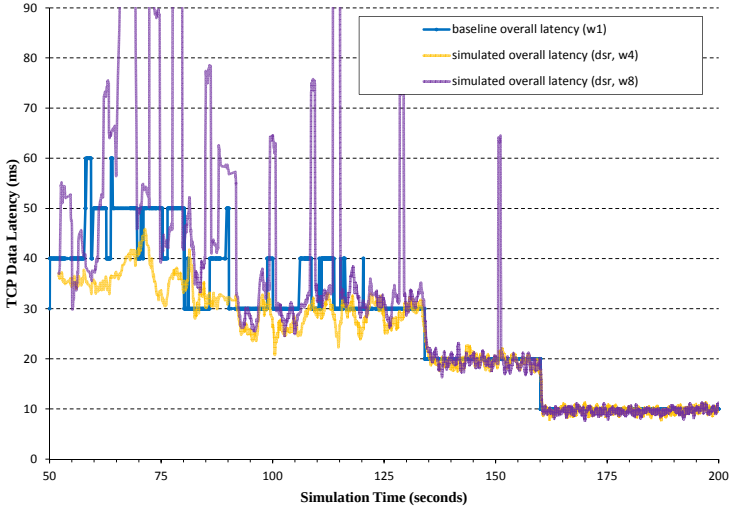
Figure 7.16 – Average TCP data latency with the *tcp_window* size parameter set to 4 for MultiWARP ordered TCP data packet arrival is shown in red dashes. The overall unordered TCP transmission with the *tcp_window* size parameter set to 4 for MultiWARP is shown in red and DSR is shown in orange. For the *tcp_window* size parameter equal to 8, MultiWARP is shown in green and DSR is shown in purple. The blue curve corresponds to the baseline overall latency (*tcp_window* size parameter of 1) multiplied by the hop-count distance between the source and destination nodes during simulation. This graph is depicted in further detail overleaf, showing the 4 data latency components in comparison to each other.



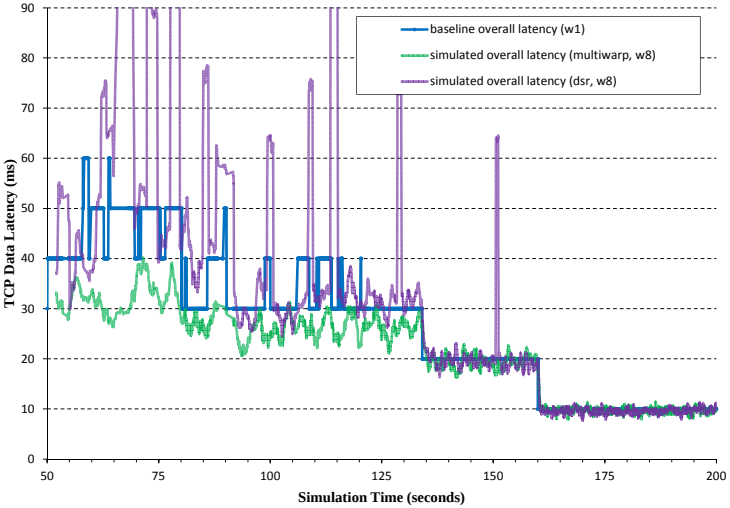
(a) MultiWARP *tcp_window* 4 , DSR *tcp_window* 4



(b) MultiWARP *tcp_window* 4 , MultiWARP *tcp_window* 8



(c) DSR *tcp_window* 4 , DSR *tcp_window* 8



(d) MultiWARP *tcp_window* 8 , DSR *tcp_window* 8

Figure 7.16(a–d) – Average TCP data latency with *tcp_window* size parameter equal to 4 for MultiWARP and DSR, as indicated.

In Figure 7.16(a–d), the expected overall latency curve (shown in blue) represents the expected latency using a *tcp_window* size parameter of 1. This result is similar to the overall TCP data packet latency (shown in green) in Figure 7.14 for comparison. It was stated that the expected overall TCP data packet latency should be decreased when multiple packets are transmitted. The simulated overall latency for MultiWARP and DSR with a *tcp_window* size parameter equal to 4 is shown in Figure 7.16(a), and is explained hereafter. It is noted that when the source and destination nodes become immediate neighbours (i.e., a hop-count of 1) during time period $t = [160..200]$ the latency is 10.0 ms. This is the same result when the simulation was performed using a *tcp_window* size parameter of 1. Similarly, for the time period $t = [135..160]$ which corresponds to a hop-count of 2, the overall latency is 20.0 ms for both cases.

With MultiWARP, the effect of multiple data packets traversing the network using different disjoint routing paths allows the TCP data packets to travel closer towards the destination node in terms of hop-count. This means that the latency incurred further away from the destination node (i.e., closer to the source node) can be reduced through parallelism. In Figure 7.16(b), increasing the *tcp_window* size parameter from 4 to 8 has resulted in a decrease in overall latency using MultiWARP, especially during the time period $t = [50..135]$. Due to MultiWARP adaptively distributing its TCP traffic load among all the alternative routing paths available, the overall latency becomes lower than the expected case for a *tcp_window* size parameter of 1. During this time period, the TCP data packets remain queued at the intermediate relay nodes until the channel becomes free (i.e., when the NAV is set to "idle"). This is the ideal situation, such that the destination node is continuously receiving TCP data packets hence lowering the overall latency. Thus, increasing the *tcp_window* size parameter is suitable for real-time traffic that demands smooth and consistent latency requirements at the destination node.

Again, the opposite effect is seen for DSR when the *tcp_window* size parameter is increased from 4 to 8, as shown in Figure 7.16(c). Severe degradation in the overall TCP latency occurs due to the same problems discussed previously, namely the 12 occasions when throughput reached zero during the time period $t = [50..160]$. Consequently, MultiWARP outperforms DSR when the *tcp_window* size parameter is set to 8 as it does not suffer from this problem, as shown in Figure 7.16(d).

7.4 SUMMARY

In summary, the performance evaluation of the proposed Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP) was established through computer simulation and theoretical analysis. The simulation environment was introduced by means of an overview of different topological designs in order to provide a baseline result for the comprehensive 50-node random waypoint topology. The simulation model was kept as close as possible to other literature in the field, and the simulation network parameters were kept fixed for each simulation. The simulation mobility model utilised both slow and fast moving nodes of between 1 and 19 ms^{-1} (i.e., 3.6 km/h to 68.4 km/h) to demonstrate suitability to both pedestrian and vehicular conditions.

The performance metrics evaluated included the packet delivery ratio and throughput, the end-to-end packet delay, packet loss, route discovery and acquisition delay, routing overhead, the impact of routing reactivity on the awareness region, route selection performance, and routing stability. These metrics were compared using different traffic sources such as UDP and TCP with various *tcp_window* sizes for both MultiWARP and DSR. The major findings are reviewed and concluded in the next chapter, and recommendations for future studies are presented thereafter.

CHAPTER 8

CONCLUSIONS

8.1 INTRODUCTION

The main objective of this study has been the development of a hybrid routing protocol suitable for a multi-hop wireless ad-hoc network, called the Multi-hop Wireless Ad-hoc Routing Protocol (MultiWARP). In achieving this objective, a number of major tasks have been successfully undertaken and a number of contributions have been made, as follows:

- The formulation of a hybrid routing protocol that provides a suitable routing path, including alternative backup routes, in an efficient and timely manner. Furthermore, the selected routing path can undergo route repair and be optimised on-the-fly if unexpected link breakages or shorter alternative routing paths can be found, respectively.
- The design and implementation of both proactive and reactive routing components that mutually exploit the routing information available by incorporating the strengths provided by each component.
- The design of an awareness region for the proactive routing component in order to prevent the "broadcast storm problem", and its associated flooding of the network by limiting the distance a routing update packet is propagated.
- The design of a route request algorithm for the reactive routing component that constructs and solves the reachability matrix for a given network topology. Through the application of the algorithm designed, a reduction of routing overhead is obtained by minimising the number of nodes that the route request packet must be transmitted to, termed "covercasting".

- An analytical study of the proposed covercasting algorithm showing that the average computational complexity is characterised as $O(\log n)$, where n is the number of nodes.
- The design and implementation of three different encoding schemes that encode vital network topology information such as node connectivity and routing metrics. Using these encoding schemes, the size of the route update packet is minimised thereby reducing routing overhead.
- Finally, the performance of the proposed hybrid routing protocol is evaluated by computer simulation to obtain the overall network throughput, packet losses, delay characteristics, route discovery latency and overhead, and routing stability characteristics.

8.2 CONCLUSIONS

In conclusion, it has been shown in Section 7.3.1 that the proposed routing protocol is able to achieve an average packet delivery ratio of 97.46% when offered a CBR traffic load, outperforming DSR's ratio of 96.62%. More importantly, the end-to-end packet delay results in Section 7.3.2 show that MultiWARP outperforms DSR by an average of 15.7%. The difference in performance is due to the selection of optimum routing paths in MultiWARP compared to DSR's selection of non-optimum paths at a given time instant. This means that users will experience a higher packet delivery ratio and a decrease in packet delay using the proposed routing protocol.

In the analysis of the route discovery and acquisition delay characteristics in Section 7.3.4, the proposed routing protocol can obtain a valid routing path between a source and destination node much faster than DSR. By exploiting the information contained within the routing table, and by using the reachability matrix and covercasting technique, discussed in Section 6.3, MultiWARP is able to discover and acquire a valid routing path in an average of 16.07 ms, compared to DSR's 540.43 ms. As a result, there is a 33.6 fold decrease in time before the data packet stored within the

packet buffer of the source node can be transmitted. The improved discovery times are due to the advantages provided by use of a hybrid routing design and the redundancy of the ARP protocol. In this way, the proposed routing protocol is able to offer a more responsive performance to users for data transmission.

The study performed in Section 7.3.5 shows that the awareness region parameter R greatly influences the reactivity behaviour of the protocol and the RUPDT packet size. Also, by transmitting a RUPDT packet every μ seconds, it is possible to avoid the "broadcast storm problem" whilst at the same time guaranteeing automatic self-configuration within a maximum of μ seconds after start-up. It is shown that a plateau in routing information is observed as R is increased, thus, by limiting the size of the awareness region a significant reduction in RUPDT packet size is achievable. This is further minimised by use of the proposed RUPDT encoding schemes, presented in Sections 5.3 and 5.4, which encode the RUPDT packet using the minimum amount of data bits.

Using the proposed routing protocol, the utilisation and distribution of the traffic on the channel is guided through the use of two secondary route selection criteria, as discussed in Section 7.3.6. The throughput and latency characteristics of both selection criteria are the same as examined in Section 7.3.7. Furthermore, it is shown in Section 7.3.8 that MultiWARP offers improved throughput and stability performance when compared to DSR when various TCP traffic loads are offered. In particular, when the *tcp_window* size parameter is increased beyond 4, the DSR protocol suffers from the TCP instability problem caused by link-failures. As the proposed routing protocol adaptively distributes the TCP traffic load among all the alternative routing paths available, it markedly outperforms DSR as it can bypass link-failures without having to deploy a route discovery procedure. As a result, users can benefit from stable throughput performance without interruptions caused by link-failures occurring between intermediate relay nodes.

8.3 RECOMMENDATIONS

In this research, the selection of the optimum routing path is based on the minimum hop-count between a source and destination node. This can be improved upon by using an alternative metric than hop-count alone, such as one that encompasses nodal velocity or utilisation statistics. In an attempt to support these features, MultiWARP does support the inclusion of using other metric information such as congestion control information and bandwidth availability using the *Options* field, as discussed in Section 5.3. If these additional metrics were used in conjunction with the computationally efficient algorithms proposed in Sections 6.2 and 6.3, it would be possible to improve distribution of the traffic load to achieve even better network performance.

APPENDIX A

MULTIWARP HEADER SPECIFICATION

A.1 MULTIWARP PACKET HEADER

To distinguish whether a data packet has a MultiWARP packet header, the use of a unique identifier must be included within the 8-bit *Protocol* field of the IPv4 packet header, as shown in Figure 2.10 in Section 2.6. This header is present in all data packets; therefore, the *Protocol* field will indicate whether a MultiWARP packet header, as shown in Figure A.1, is encapsulated within the *Options* field of the IPv4 packet header. It should be noted that the original 8-bit *Protocol* field of the IPv4 packet header is stored in the *Protocol* field of the MultiWARP packet header, and thus the original value can be restored at the destination node.

As discussed in Section 2.6, the *Options* field can be used to specify additional packet headers before the *Data* component of the packet. The *Data* component primarily consists of higher-layer protocol headers and binary data that are ignored at the routing layer. As a result, the MultiWARP packet header resides between the IP layer (OSI layer 3) and TCP/UDP layer (OSI layer 4), as shown in Table 2.1 and Table 2.2 in Section 2.3.

The IPv4 packet header specifies a 4-bit *Header Length* field to indicate the number of 32-bit words comprising the entire IPv4 packet header including all relevant *Options* before the *Data* component begins. Therefore, the maximum number of 32-bit words that can be allocated is $\left(2^4 - 1\right) = 15$, or alternatively 60 bytes. Given that the IPv4 packet header itself consumes 160 bits, or 20 bytes, it leaves a remainder of 320 bits, or 40 bytes, for the *Options* field. This is an insufficient number of bytes for storage, especially for route update (RUPDT) packets. As a result, the *Header Length* field should only specify the length of the IPv4 packet header plus the minimum 32-bits for the *Options* field to store the MultiWARP packet header. This

thereby excludes the payload of the MultiWARP packet, namely, the *Internal Packet Structure*, from the *Header Length* field. Instead, the *Data* field of the IPv4 header will point to the *Internal Packet Structure*, and not the higher-layer *Data* contained within it if such *Data* is present (e.g., in the case of a source-routed data packet).

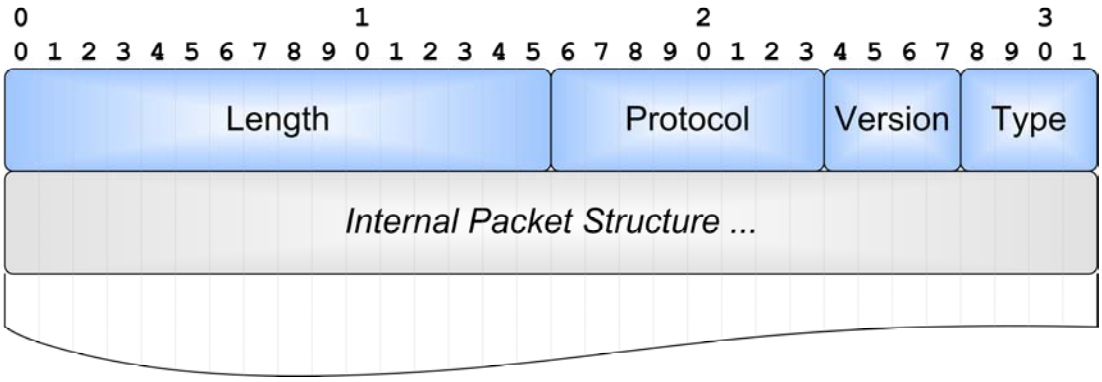


Figure A.1 – MultiWARP Packet Header structure defined within the IPv4 *Options* field.

The 32-bit MultiWARP packet header is comprised of four fields to allow the routing protocol to correctly determine the function of the MultiWARP packet. In effect, it determines the MultiWARP packet type, thereby informing the routing protocol how to handle the information contained within the *Internal Packet Structure*.

Length: This 16-bit field indicates the total number of bytes immediately following the MultiWARP packet header, and is used to specify the length of data storage required for a particular MultiWARP packet type. Given the 16-bit limitation, the maximum number of bytes permissible is therefore $\left(2^{16}-1\right) = 65535$ bytes.

Protocol: This 8-bit field is used to store the original 8-bit value specified within the *Protocol* field of the IPv4 packet header. This is due to the *Protocol* field of the IPv4 packet header being overwritten by the MultiWARP unique identifier to indicate that it contains a MultiWARP packet header. Furthermore, it allows the destination node to restore the *Protocol* field of the IPv4 packet header, and remove the MultiWARP packet header structure before passing the received packet up to the TCP/UDP (OSI layer 4) handler.

Version: This 4-bit field is used to indicate the revision history of the MultiWARP routing protocol, and should be set to zero to indicate revision 1.0 of the protocol. All other combinations are reserved for future use, and should not be set.

Type: This 4-bit field is used to indicate the type of MultiWARP packet that is being carried in the *Internal Packet Structure* in order for the routing protocol to correctly interpret its function. Given the 4-bit limitation, this allows for a maximum of $\left(2^4\right)=16$ different MultiWARP packet types to be defined. The different types are shown in Table A.1, and are discussed in Sections A.2–A.8, respectively.

Type	Description
0000	Routing Path (RP) Header
0001	Route Update (RUPDT) Header
0010	Route Request (RREQ) Header
0011	Route Reply (RREP) Header
0100	Route Error (RERR) Header
0101	SuperNode Request (SNREQ) Header
0110	SuperNode Reply (SNREP) Header

Table A.1 – Type field binary patterns and its corresponding description.

Internal Packet Structure: This field is a placeholder and is not part of the 32-bit MultiWARP packet header, but is illustrated to indicate the location of the MultiWARP packet payload. The number of bytes indicated by the *Length* field is appended after the *Type* field to provide storage for the data pertaining to the type of MultiWARP packet, as discussed in Sections A.2–A.8.

A.2 ROUTING PATH (RP) HEADER

For a packet to be acceptable for transmission, the routing path must be entirely specified from the source node to the destination node (i.e., the source-route). The routing path is obtained by searching the local routing table that is provided by the

proactive and reactive components of the routing protocol. The best routing path according to certain selection criteria is then selected, and included in the RP header.

To provide storage for the source-route, the MultiWARP packet header must have the *Type* field set to '0' (binary 0000) to indicate an RP packet type. In addition, the *Length* field of the MultiWARP packet header should be set to the size of the internal structure of the RP header, as shown in Figure A.2.

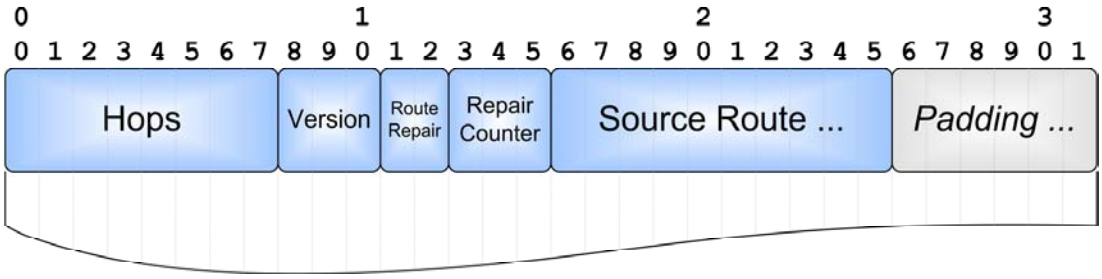


Figure A.2 – Routing Path (RP) Header defined within the *Internal Packet Structure* field of the MultiWARP Packet Header.

The internal structure of the RP header is comprised of the following six fields to allow the routing protocol to utilise the source-route and deliver the packet to its destination.

Hops: This 8-bit field indicates the number of hops used in the source-route, including the source node, the destination node, and all the intermediate relay nodes in between. Using this field, it is possible to determine the number of bits used to represent the node address by dividing the *Length* field of the MultiWARP packet header by the number of *Hops* specified in the RP header. Taking into account the 16-bits (2 bytes) consumed by the first four fields, the *Address_bit_length* variable can be calculated using equation A.1(a).

$$Address\_bit\_length = \left[\frac{(Length - 2)}{Hops} \right] \times 8 \qquad \text{(Eq. A.1a)}$$

For example, if the *Length* field within the MultiWARP packet header is indicated as 50-bytes, and the *Hops* field in the RP header is indicated as 12, it can be concluded that 32-bit IPv4 addressing is used, as given by equation A.1(b).

$$Address\_bit\_length = \left[\frac{(50-2)}{12} \right] \times 8 = 32 \qquad \text{(Eq. A.1b)}$$

Alternatively, if the *Hops* field was equal to 3, it can be concluded that 128-bit IPv6 addressing is used, as given by equation A.1(c).

$$Address\_bit\_length = \left[\frac{(50-2)}{3} \right] \times 8 = 128 \qquad \text{(Eq. A.1c)}$$

As a result, the structure of the MultiWARP packet header, and accordingly the RP header, does not need to be modified in order to support different future addressing schemes such as the IPv6 standard.

Version: This 3-bit field is used to indicate the version of the RP header, and should be set to zero to indicate revision 1.0 of the protocol. All other combinations are reserved for future use, and should not be set. For future versions, this field can be used to indicate that additional information is available for selected or all nodes specified in the *Source Route* field, and is dependent of the actual scheme utilised.

Route Repair: This 2-bit field specifies whether the routing path specified in the *Source Route* field is allowed to be repaired and/or optimised, as discussed in Section 6.6.1.

Repair Counter: This 3-bit field indicates the number of times the routing path specified in the *Source Route* field has been repaired and/or optimised. This is required in order to prevent infinite route repairs occurring by intermediate relay nodes, and is limited to $\left(2^3 - 1\right) = 7$ repairs before the packet is dropped and a RERR packet is generated, as discussed in Section 6.6.1.

Source Route: This variable length field specifies the sequential list of unique node addresses starting with the source node, and each intermediate relay node through

which the packet must traverse, and terminating with the destination node. The length of this field is given by the *Address_bit_length* field, as calculated from the *Hops* field discussed above. The uniqueness of the addresses inherently guarantees loop-free routing. The minimum number of addresses contained within the *Source Route* field is 2 (i.e., the source node and destination node) and maximum is indicated by the *Hops* field.

Padding: This variable length field is used on the tail of the completed RP header, and is padded with '0' bits to the nearest byte for alignment purposes.

A.3 ROUTE UPDATE (RUPDT) HEADER

The route update (RUPDT) packet is crucial in the task of disseminating routing path information to neighbouring nodes within the network. The RUPDT packet is transmitted periodically by each node every μ seconds by default, unless otherwise specified in the *Options* field, as discussed in Section 5.2.3. The RUPDT packet must contain the entire topology for up to a distance of $(R-1)$ hops away from the source node that is transmitting the packet. Before the RUPDT is constructed from the information in the local routing table, a route maintenance routine should be executed to remove any stale nodes, as discussed in Section 5.2.3.

The minimum required number of fields that must be transmitted for each node that the source node is aware of includes the *Node Address*, *Node Sequence Number*, and the *Connectivity Matrix* depicting the connectivity between the nodes. Any other fields such as the RUPDT periodic update interval μ, or routing metric fields such as congestion control information, and/or bandwidth availability, etc., can be specified in the *Options* field for each node, and is described in further detail below.

To provide storage for the topological connectivity information, the MultiWARP packet header must have the *Type* field set to '1' (binary 0001) to indicate an RUPDT packet type. In addition, the *Length* field should be set to the size of the internal structure of the RUPDT header, as shown in Figure A.3.

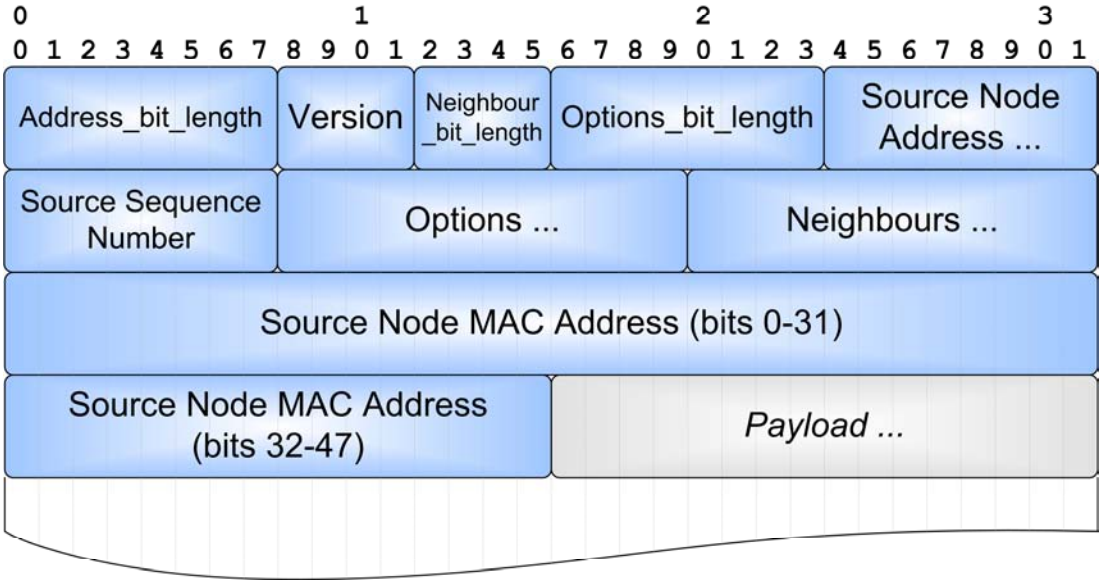


Figure A.3 – Route Update (RUPDT) Header defined within the *Internal Packet Structure* field of the MultiWARP Packet Header.

The internal structure of the RUPDT header is comprised of the following nine fields to allow the routing protocol to disseminate valid routing paths through the proactive component.

Address_bit_length: This 8-bit field indicates the number of bits (plus 1) that comprises the bit length of the node addresses. For example, a value equal to 31 would indicate 32-bit IPv4 addressing. Given the 8-bit limitation, the maximum address length allowed is therefore $\left(2^8\right) = 256$ bits, thus supporting the IPv6 addressing mode which only requires 128 bits. It should be noted that this field is not related to the calculated *Address_bit_length* variable used in the routing path (RP) header.

Version: This 4-bit field is used to indicate the version of the RUPDT header, and should be set to zero to indicate revision 1.0 of the protocol using the proposed encoding scheme presented in Section 5.3. All other combinations are reserved for future use, and should not be set. For future versions, this field can be used to indicate a different encoding scheme being used to encode the route update (RUPDT) payload structure, or the structure of any *Options* fields, as discussed in Section 5.4.

Neighbour_bit_length: This 4-bit field indicates the number of bits required to encode the *Neighbours* field. The number of bits allowed to encode the *Neighbours* field is therefore between 0 and $\left(2^4 - 1\right) = 15$ bits. As a result, each node can theoretically support a maximum of $\left(2^{15} - 1\right) = 32767$ neighbouring nodes.

Options_bit_length: This 8-bit field indicates the number of bits required to encode the *Options* field. This allows for a maximum of $\left(2^8 - 1\right) = 255$ bits to be used as an optional data structure for each node to include additional information. If the value is set to zero, no optional data structure is provided, and therefore the *Options* field becomes non-existent in the RUPDT packet.

Source Node Address: This variable length field specifies the address of the node that is transmitting the RUPDT packet, i.e., the source node. The length of this field is given by the *Address_bit_length* field, as discussed above.

Source Sequence Number: This 8-bit field indicates the sequence number of the node that is transmitting the RUPDT packet, i.e., the source node. The value is incremented by 1 before each subsequent transmission of the RUPDT packet, and is circular in nature (excluding the reserved value 0), as discussed in Section 5.2.3.

Options: This variable length field specifies the data bits used to represent any optional information for the source node, such as the RUPDT periodic update interval μ, or routing metric fields such as congestion control information, and/or bandwidth availability, etc. The length of this field is given by the *Options_bit_length* field, as discussed above. The structure of the *Options* field can be arbitrarily defined for any given implementation, and must be ignored by all other implementations, as indicated by the *Version* field in the RUPDT header.

Neighbours: This variable length field specifies the number of nodes present in the proceeding segment 1 hop away from the source node (i.e., the immediate neighbours). The length of this field is given by the *Neighbour_bit_length* field, as discussed above.

Source Node MAC Address: This 48-bit field specifies the MAC address of the node that is transmitting the RUPDT packet, i.e., the source node. The addition of the MAC address results in the redundancy of the ARP protocol, as discussed in Section 5.5.

Payload: This field is a placeholder and is not part of the RUPDT header, but is illustrated to indicate the location of the RUPDT payload structure. The *Payload* field is a structure comprised of six fields, as discussed in Section A.3.1.

A.3.1 Route Update (RUPDT) Payload Structure

For the RUPDT encoding scheme discussed in Section 5.3, there are $(R-1)$ consecutive *Payload* segments to accommodate all the nodes of a similar hop-count group, ranging from 1 to $(R-1)$ hops. Each consecutive *Payload* segment has δ node entries, where δ is the value given by the previous segment's *Neighbours* field. For the first segment, this value is given in the *Neighbours* field of the RUPDT header. Each of the δ node entries consist of the first 3 fields of the RUPDT payload structure; namely *Node Address*, *Node Sequence Number*, and the *Options* field if enabled in the RUPDT header. After the δ node entries are specified, it is then followed by the *Neighbours* field (which is used by the next *Payload* segment to obtain the value of δ), and the current *Payload* segment's *Connectivity Matrix*. The RUPDT payload structure for the RUPDT encoding scheme discussed in Section 5.3 is shown in Figure A.3.

For the two alternative RUPDT encoding schemes presented in Sections 5.4.1–5.4.2, refer to Figure 5.14 and Figure 5.16 for the RUPDT payload structure, respectively.

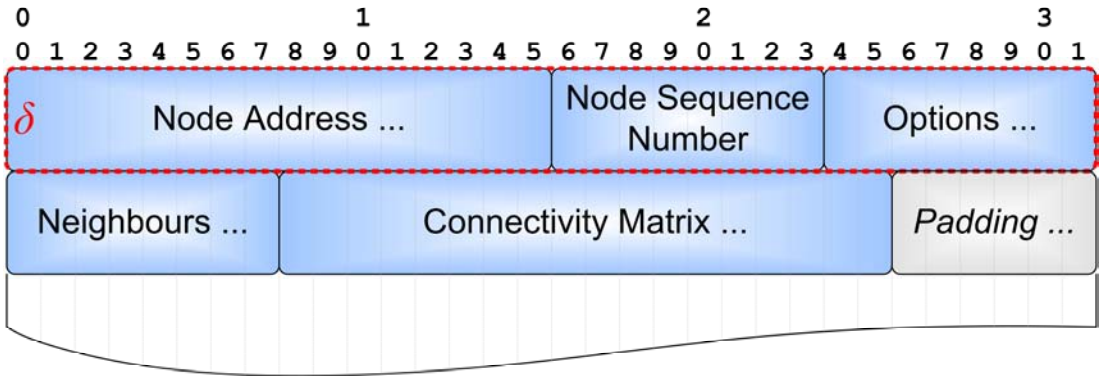


Figure A.4 – Route Update (RUPDT) Payload structure defined within the RUPDT Header.

The internal structure of the RUPDT payload is comprised of the following six fields, and contains the topological information needed to depict the connectivity between nodes up to $(R-1)$ hops away from the source node.

Node Address: This variable length field specifies the node address of the node at a certain hop-count within the *Payload* segment. The length of this field is given by the *Address_bit_length* field, as discussed above.

Node Sequence Number: This 8-bit field indicates the sequence number of the node at a certain hop-count within the *Payload* segment. The value is not incremented or modified in any way, but simply transmitted as is.

Options: This variable length field specifies the data bits used to represent any optional information for the node, such as the RUPDT periodic update interval μ, or routing metric fields such as congestion control information, and/or bandwidth availability, etc. The length of this field is given by the *Options_bit_length* field, as discussed above. The structure of the *Options* field can be arbitrarily defined for any given implementation, and must be ignored by all other implementations, as indicated by the *Version* field in the RUPDT header.

Neighbours: This variable length field specifies the number of nodes present in the proceeding *Payload* segment 1 hop away from the current *Payload* segment. The length of this field is given by the *Neighbour_bit_length* field, as discussed above.

Connectivity Matrix: This variable length field specifies the connectivity between the δ node entries in the current *Payload* segment and the node entries from the previous *Payload* segment. The connectivity is indicated by a single bit, '0' indicating no connection, and '1' indicating a bi-directional link. The length of this field is given by $(X \times Y)$ bits, where Y is the value of the previous *Payload* segment's *Neighbours* field, and X is the value of the *Payload* segment's *Neighbours* field previous to Y. It should be noted that the first segment is always (1×1) bits.

Padding: This variable length field is used on the tail of the completed RUPDT payload structure, and is padded with '0' bits to the nearest byte for alignment purposes. It is only used for the final *Payload* segment, and is not used to pad all of the $(R-1)$ consecutive *Payload* segments.

A.4 ROUTE REQUEST (RREQ) HEADER

The route request (RREQ) packet is a reactively generated packet created during the route discovery process, and covercasted to the selected candidate nodes through a series of unicast transmissions. The RREQ packet determines the routing path between the source node and the destination node by accumulating the routing path information as it traverses between candidate nodes via intermediate relay nodes. As discussed in Section 6.5, the covercasting method only requires the set of candidate nodes of the previous candidate node to be explicitly defined within the packet. This is because the remaining nodes will be implicitly removed due to the exploitation of the topological knowledge already available in the local routing table by the subsequent candidate node.

To provide the storage for the source-route and covercasting information, the MultiWARP packet header must have the *Type* field set to '2' (binary 0010) to indicate an RREQ packet type. In addition, the *Length* field should be set to the size of the internal structure of the RREQ header, as shown in Figure A.5.

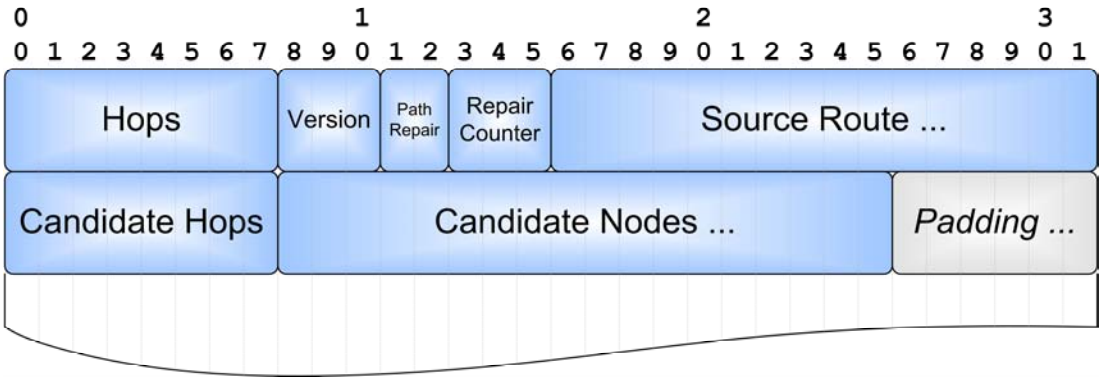


Figure A.5 – Route Request (RREQ) Header defined within the *Internal Packet Structure* field of the MultiWARP Packet Header.

The internal structure of the RREQ header is comprised of the following eight fields to allow the routing protocol to accumulate a valid routing path between the source node and the destination node that lies outside of the awareness region. The first five fields of the RREQ header are the same as the RP header, as the packet requires a source-route to reach the selected candidate node. As such, these five fields are defined in Section A.2. The remaining three fields that are required for the covercasting process are given below.

Candidate Hops: This 8-bit field indicates the number of candidate node addresses specified in the *Candidate Nodes* field.

Candidate Nodes: This variable length field specifies the list of candidate node addresses, taking into account the forking node removal problem, as discussed in Section 6.5.1. The length of this field is given by the *Address_bit_length* field, as calculated from the *Hops* field of the RP header, as discussed in Section A.2.

Padding: This variable length field is used on the tail of the completed RREQ header, and is padded with '0' bits to the nearest byte for alignment purposes.

A.5 ROUTE REPLY (RREP) HEADER

The route reply (RREP) packet is transmitted by a candidate node back to the source node that originally initiated the route discovery process. It should be noted that an intermediate relay node can also reply with a RREP packet back to the source node, not just the selected candidate nodes. The routing path determined by the route discovery process is then reversed using a route reversal technique, which consists of reversing the traversed route comprised of intermediate relay nodes back to the source.

To provide the storage for the source-route information, the MultiWARP packet header must have the *Type* field set to '3' (binary 0011) to indicate an RREP packet type. In addition, the *Length* field should be set to the size of the internal structure of the RREP header, as shown in Figure A.6.

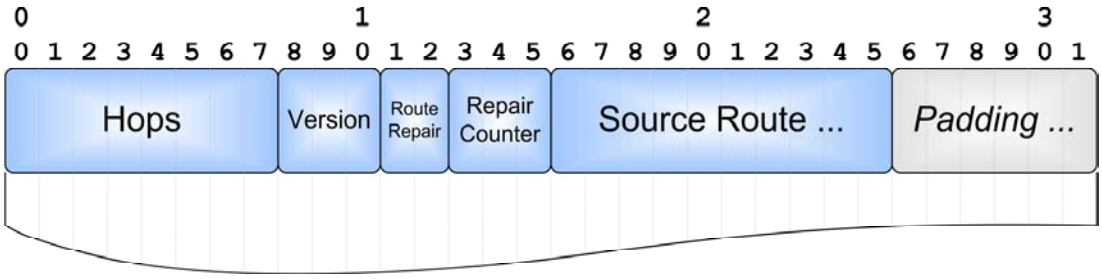


Figure A.6 – Route Reply (RREP) Header defined within the *Internal Packet Structure* field of the MultiWARP Packet Header.

The internal structure of the RREP header is identical to that of the RP header, as the packet requires a source-route to reach the source node, with the only difference being the *Type* field in the MultiWARP packet header. As such, the six fields are the same as the fields defined for the RP header, as discussed in Section A.2. Furthermore, it becomes possible to attach the RREP to an outgoing data packet from the current node if it is destined to the source node. This is because the RREP header behaves like an RP header. The length of the *Data* included in the packet can be determined from the *Length* field in the IPv4 header.

A.6 ROUTE ERROR (RERR) HEADER

The route error (RERR) packet is transmitted by an intermediate relay node back to the source node that originated the erroneous source-route route by reversing the traversed route, indicating the segment where the failure occurred.

To provide the storage for the offending node address, the MultiWARP packet header must have the *Type* field set to '4' (binary 0100) to indicate an RERR packet type. In addition, the *Length* field should be set to the size of the internal structure of the RERR header, as shown in Figure A.7.

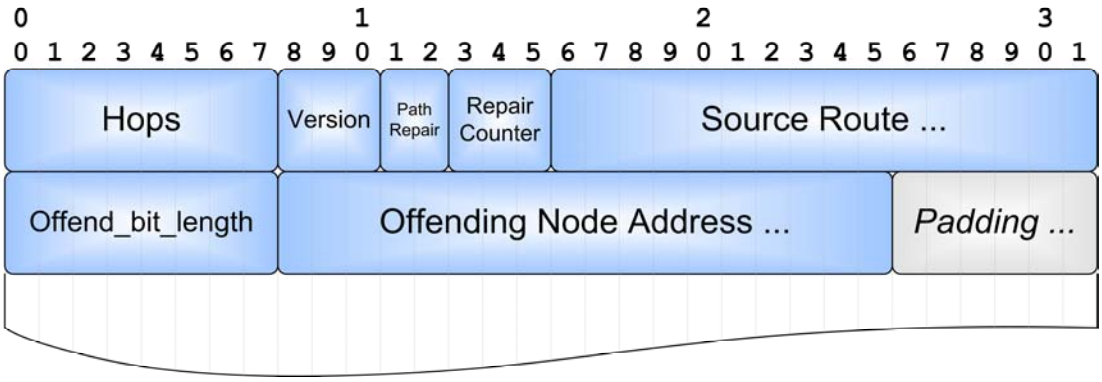


Figure A.7 – Route Error (RERR) Header defined within the *Internal Packet Structure* field of the MultiWARP Packet Header.

The internal structure of the RERR header is comprised of the following eight fields to allow the neighbouring nodes to mark the offending node by setting the offending node to the inactive "expired" state in their local routing tables. The first five fields of the RERR header are the same as the RP header, as the packet requires a source-route to reach the source node. As such, these five fields are defined in Section A.2. The remaining three fields that are required to indicate the offending node address are given below.

Offend_bit_length: This 8-bit field indicates the number of bits (plus 1) that comprises the length of the node addresses. For example, a value equal to 31 would indicate 32-bit IPv4 addressing. Given the 8-bit limitation, the maximum address

length allowed is therefore $\left(2^8\right) = 256$ bits, thus supporting the IPv6 addressing mode which only requires 128 bits.

Offending Node Address: This variable length field specifies the node address of the offending node that was unreachable by the intermediate relay node, as obtained from the *Source Route* field of the RP header. The length of this field is given by the *Address_bit_length* field, as discussed above. The broken segment is therefore located between the node transmitting the RERR packet and the *Offending Node Address*.

Padding: This variable length field is used on the tail of the completed RERR header, and is padded with '0' bits to the nearest byte for alignment purposes.

A.7 SUPERNODE REQUEST (SNREQ) HEADER

The SuperNode request (SNREQ) packet is transmitted by certain nodes in order to maintain an extended routing table by requesting an encapsulated RUPDT packet from each of the selected candidate nodes, as discussed in Section 5.6.1. By obtaining the routing table information from each of the selected candidate nodes, it gives the largest possible increase in topological information, resulting in a larger awareness region of $\left(2R-1\right)$ hops.

To provide storage for the source-route to the selected candidate node, the MultiWARP packet header must have the *Type* field set to '5' (binary 0101) to indicate an SNREQ packet type. In addition, the *Length* field of the MultiWARP packet header should be set to the size of the internal structure of the SNREQ header, as shown in Figure A.8.

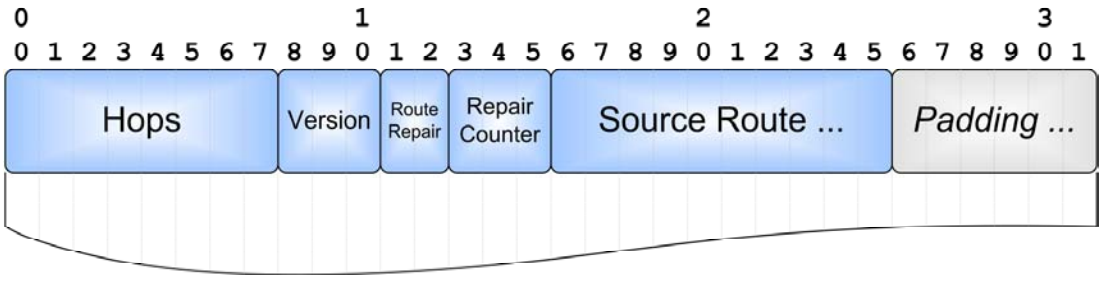


Figure A.8 – SuperNode Request (SNREQ) Header defined within the *Internal Packet Structure* field of the MultiWARP Packet Header.

The internal structure of the SNREQ header is identical to that of the RP header, as the packet requires a source-route to reach the selected candidate node, with the only difference being the *Type* field in the MultiWARP packet header. As such, the six fields are the same as the fields defined for the RP header, as discussed in Section A.2. Furthermore, it becomes possible to attach the SNREQ to an outgoing data packet from the current node to the selected candidate node, as the SNREQ header behaves like an RP header. The length of the *Data* included in the packet can be determined from the *Length* field in the IPv4 header.

A.8 SUPERNODE REPLY (SNREP) HEADER

The SuperNode reply (SNREP) packet is transmitted by a candidate node back to the source node that requested the extended routing table information. The RUPDT packet created at the candidate node is then encapsulated within the SNREP packet using the *Data* field of the IPv4 packet. This results in a larger awareness region of $(2R-1)$ hops for the node that originated the SNREQ packet. The routing path contained within the SNREQ packet is then reversed using a route reversal technique, which consists of reversing the traversed route comprised of intermediate relay nodes back to the requesting node.

To provide storage for the source-route to the requesting node, the MultiWARP packet header must have the *Type* field set to ‘6’ (binary 0110) to indicate an SNREP packet type. In addition, the *Length* field of the MultiWARP packet header should

be set to the size of the internal structure of the SNREP header, as shown in Figure A.9.

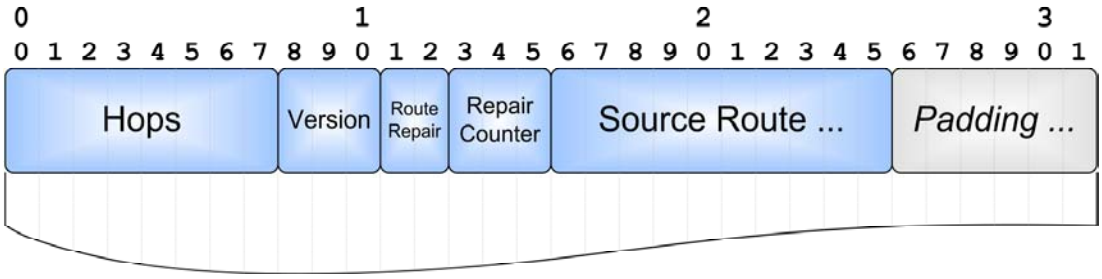


Figure A.9 – SuperNode Reply (SNREP) Header defined within the *Internal Packet Structure* field of the MultiWARP Packet Header.

The internal structure of the SNREP header is identical to that of the RP header, as the packet requires a source-route to reach the node requesting the extended routing table information, with the only difference being the *Type* field in the MultiWARP packet header. As such, the six fields are the same as the fields defined for the RP header, as discussed in Section A.2. Regardless of the RUPDT encoding scheme used, as discussed in Section 5.3, upon reception of the SNREP packet by the requesting node, the SNREP header is removed and the *Data* encapsulated within the IPv4 packet it is processed as if it received an RUPDT packet, with the exception of including nodes up to $(2R-1)$ hops away as opposed to $(R-1)$ hops.

APPENDIX B

VALID REACHABILITY MATRIX COMBINATIONS

All possible valid combinations for a (1×1) reachability matrix:

bits:0 total:1

```
Cover Matrix:
1
```

All possible valid combinations for a (2×2) reachability matrix:

bits:1 total:2

```
Cover Matrix:
1 1
1 1
```

```
Cover Matrix:
1 0
0 1
```

All possible valid combinations for a (3×3) reachability matrix:

bits:3 total:8

```
Cover Matrix:
1 0 1
0 1 0
1 0 1
```

```
Cover Matrix:
1 1 1
1 1 0
1 0 1
```

```
Cover Matrix:
1 0 0
0 1 1
0 1 1
```

```
Cover Matrix:
1 1 0
1 1 1
0 1 1
```

```
Cover Matrix:
1 0 1
0 1 1
1 1 1
```

```
Cover Matrix:
1 1 1
1 1 1
1 1 1
```

```
Cover Matrix:
1 1 0
1 1 0
0 0 1
```

```
Cover Matrix:
1 0 0
0 1 0
0 0 1
```

All possible valid combinations for a (4×4) reachability matrix:

bits:6 total:64

```
Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 0 1             1 0 1 0             1 0 0 1             1 1 0 0
0 1 0 0             0 1 0 0             0 1 1 1             1 1 0 1
0 0 1 0             1 0 1 1             0 1 1 1             0 0 1 0
1 0 0 1             0 0 1 1             1 1 1 1             0 1 0 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 0 0             1 0 1 1             1 0 1 0             1 1 0 1
0 1 0 1             0 1 0 0             0 1 1 0             1 1 0 1
0 0 1 0             1 0 1 1             1 1 1 0             0 0 1 0
0 1 0 1             1 0 1 1             0 0 0 1             1 1 0 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 0 1             1 0 1 0             1 0 1 1             1 1 0 0
0 1 0 1             0 1 0 1             0 1 1 0             1 1 0 0
0 0 1 0             1 0 1 1             1 1 1 0             0 0 1 1
1 1 0 1             0 1 1 1             1 0 0 1             0 0 1 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 0 0             1 0 1 1             1 0 1 0             1 1 0 1
0 1 0 0             0 1 0 1             0 1 1 1             1 1 0 0
0 0 1 1             1 0 1 1             1 1 1 0             0 0 1 1
0 0 1 1             1 1 1 1             0 1 0 1             1 0 1 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 0 1             1 0 0 0             1 0 1 1             1 1 0 0
0 1 0 0             0 1 1 0             0 1 1 1             1 1 0 1
0 0 1 1             0 1 1 0             1 1 1 0             0 0 1 1
1 0 1 1             0 0 0 1             1 1 0 1             0 1 1 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 0 0             1 0 0 1             1 0 1 0             1 1 0 1
0 1 0 1             0 1 1 0             0 1 1 0             1 1 0 1
0 0 1 1             0 1 1 0             1 1 1 1             0 0 1 1
0 1 1 1             1 0 0 1             0 0 1 1             1 1 1 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 0 1             1 0 0 0             1 0 1 1             1 1 1 0
0 1 0 1             0 1 1 1             0 1 1 0             1 1 0 0
0 0 1 1             0 1 1 0             1 1 1 1             1 0 1 0
1 1 1 1             0 1 0 1             1 0 1 1             0 0 0 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 1 0             1 0 0 1             1 0 1 0             1 1 1 1
0 1 0 0             0 1 1 1             0 1 1 1             1 1 0 0
1 0 1 0             0 1 1 0             1 1 1 1             1 0 1 0
0 0 0 1             1 1 0 1             0 1 1 1             1 0 0 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 1 1             1 0 0 0             1 0 1 1             1 1 1 0
0 1 0 0             0 1 1 0             0 1 1 1             1 1 0 1
1 0 1 0             0 1 1 1             1 1 1 1             1 0 1 0
1 0 0 1             0 0 1 1             1 1 1 1             0 1 0 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 1 0             1 0 0 1             1 1 0 0             1 1 1 1
0 1 0 1             0 1 1 0             1 1 0 0             1 1 0 1
1 0 1 0             0 1 1 1             0 0 1 0             1 0 1 0
0 1 0 1             1 0 1 1             0 0 0 1             1 1 0 1

Cover Matrix:       Cover Matrix:       Cover Matrix:       Cover Matrix:
1 0 1 1             1 0 0 0             1 1 0 1             1 1 1 0
0 1 0 1             0 1 1 1             1 1 0 0             1 1 0 0
1 0 1 0             0 1 1 1             0 0 1 0             1 0 1 1
1 1 0 1             0 1 1 1             1 0 0 1             0 0 1 1
```

(continued)

Cover Matrix:
1 1 1 1
1 1 0 0
1 0 1 1
1 0 1 1

Cover Matrix:
1 1 0 0
1 1 1 1
0 1 1 0
0 1 0 1

Cover Matrix:
1 1 0 1
1 1 1 1
0 1 1 1
1 1 1 1

Cover Matrix:
1 1 1 0
1 1 1 0
1 1 1 1
0 0 1 1

Cover Matrix:
1 1 1 0
1 1 0 1
1 0 1 1
0 1 1 1

Cover Matrix:
1 1 0 1
1 1 1 1
0 1 1 0
1 1 0 1

Cover Matrix:
1 1 1 0
1 1 1 0
1 1 1 0
0 0 0 1

Cover Matrix:
1 1 1 1
1 1 1 0
1 1 1 1
1 0 1 1

Cover Matrix:
1 1 1 0
1 1 1 1
1 0 1 1
0 1 1 1

Cover Matrix:
1 1 1 1
1 1 0 1
1 0 1 1
1 1 1 1

Cover Matrix:
1 1 0 0
1 1 1 0
0 1 1 1
0 0 1 1

Cover Matrix:
1 1 1 1
1 1 1 0
1 1 1 0
1 0 0 1

Cover Matrix:
1 1 1 0
1 1 1 1
0 1 1 0
1 0 1 1

Cover Matrix:
1 1 1 1
1 1 1 1
1 1 1 0
0 1 0 1

Cover Matrix:
1 1 1 1
1 1 1 0
1 1 1 1
0 1 1 1

Cover Matrix:
1 1 0 1
1 1 1 0
0 1 1 0
1 0 0 1

Cover Matrix:
1 1 0 0
1 1 1 1
0 1 1 1
0 1 1 1

Cover Matrix:
1 1 1 1
1 1 1 1
1 1 1 0
1 1 0 1

Cover Matrix:
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1

REFERENCES

3GPP 2003, *GSM/EDGE Radio Access Network (GERAN) overall description; Stage 2 (TS 43.051)*, 3rd Generation Partnership Project (3GPP).

Adibi, S. & Agnew, G.B. 2008, 'Multilayer flavoured dynamic source routing in mobile ad-hoc networks', *Communications, IET* vol. 2, no. 5, pp. 690-707.

Agrawal, D.P., Gossain, H., Cavalcanti, D. & Mohapatra, P. 2008, 'Recent advances and evolution of WLAN and WMAN standards', *Wireless Communications, IEEE,* vol. 15, no. 5, pp. 54-5.

Ahn, S. & Udaya-Shankar, A. 2001, 'Adapting to route-demand and mobility (ARM) in ad hoc network routing', *Network Protocols, 2001. Ninth International Conference on*, pp. 44-52.

Arslan, H., Chen, Z.N. & Di Benedetto, M.-G. 2006, *Ultra Wideband Wireless Communication*, Wiley-Interscience Publications, New Jersey.

Awdeh, R.Y. 2007, 'Compatibility of TCP Reno and TCP Vegas in wireless ad hoc networks', *Communications, IET,* vol. 1, no. 6, pp. 1187-94.

Baldoni, R. & Beraldi, R. 2001, 'Low Cost Routing in Mobile Ad-hoc Networks: Is It Achievable? ', in *Proceedings of the 8th IEEE Workshop on Future Trends of Distributed Computing Systems* IEEE Computer Society, p. 105

Ballew, S.M. 1997, *Managing IP Networks with Cisco Routers,* 1st edn, O'Reilly & Associates.

Bansal, S., Shorey, R., Chugh, S., Goel, A., Kumar, K. & Misra, A. 2002, 'The capacity of multi-hop wireless networks with TCP regulated traffic', *Global Telecommunications Conference, 2002. GLOBECOM '02. IEEE*, pp. 133-7 vol.1.

Bellman, R. 1958, 'On a Routing Problem', *Quarterly of Applied Mathematics,* vol. 16, no. 1, pp. 87-90.

Belrose, J.S. 2002, 'Reginald Aubrey Fessenden and the Birth of Wireless Telephony', *IEEE Antennas and Propagation Magazine,* vol. 44, no. 2, pp. 38-47.

Beraldi, R. & Baldoni, R. 2003, 'A caching scheme for routing in mobile ad hoc networks and its application to ZRP', *Computers, IEEE Transactions on,* vol. 52, no. 8, pp. 1051-62.

Bettstetter, C., Resta, G. & Santi, P. 2003, 'The Node Distribution of the Random Waypoint Mobility Model for Wireless Ad Hoc Networks', *IEEE Transactions on Mobile Computing,* vol. 2, no. 3, pp. 257-69.

Blum, A. & Sleator, D. 2000, 'Approximation algorithms', *CMU 15-451 (Algorithms), Fall 2000*, Carnegie Mellon University School of Computer Science.

Boukerche, A., Fabbri, A. & Das, S.K. 2000, 'Analysis of randomized congestion control in DSDV routing', *Modeling, Analysis and Simulation of Computer and Telecommunication Systems, 2000. Proceedings. 8th International Symposium on*, pp. 65-72.

Boukerche, A. & Rogers, S. 2001, 'GPS query optimization in mobile and wireless networks', *Computers and Communications, 2001. Proceedings. Sixth IEEE Symposium on*, pp. 198-203.

Braden, R. 1989, *RFC1122: Requirements for Internet Hosts - Communication Layers*.

Broch, J., Johnson, D.B. & Maltz, D.A. 1998, 'The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks', *IETF MANET Working Group, Internet Draft, draft-ietf-manet-dsr-1.txt,* vol. Internet Draft.

Broch, J., Maltz, D.A., Johnson, D.B., Hu, Y.C. & Jetcheva, J. 1998, 'A performance comparison of multi-hop wireless ad hoc network routing protocols', in *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, ACM Press, Dallas, Texas, United States, pp. 85-97.

Callon, R. 1990, 'RFC1195: Use of OSI IS-IS for Routing in TCP/IP and Dual Environments', *Internet Engineering Task Force, Network Working Group*.

Castaneda, R. & Das, S.R. 1999, 'Query localization techniques for on-demand routing protocols in ad hoc networks', in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, ACM Press, Seattle, Washington, United States, pp. 186-94.

Castaneda, R., Das, S.R. & Marina, M.K. 2002, 'Query localization techniques for on-demand routing protocols in ad hoc networks', *Wireless Networking,* vol. 8, no. 2/3, pp. 137-51.

Chaplot, A. 2002, 'A simulation study of multi-hop wireless network', *Personal Wireless Communications, 2002 IEEE International Conference on*, pp. 86-9.

Chartrand, G. 1984, *Introductory Graph Theory*, Dover Publications, New York.

Chen, J., Druschel, P. & Subramanian, D. 1999, 'A new approach to routing with dynamic metrics', *INFOCOM '99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, pp. 661-70 vol.2.

Cheney, M. 2001, *Tesla: Man Out of Time*, Touchstone Publishers, New York.

Cheng, H. & Cao, J. 2008, 'A design framework and taxonomy for hybrid routing protocols in mobile Ad Hoc networks', *Communications Surveys & Tutorials, IEEE,* vol. 10, no. 3, pp. 62-73.

Cheng, Z. & Heinzelman, W.B. 2003, 'Flooding strategy for target discovery in wireless networks', in *Proceedings of the 6th ACM international workshop on Modeling analysis and simulation of wireless and mobile systems*, ACM Press, San Diego, CA, USA, pp. 33-41.

Chvatal, V.A. 1979, 'A Greedy Heuristic for the Set-Covering Problem', *Mathematics of Operations Research,* vol. 4, no. 3, pp. 233-5.

Cisco Systems 2005a, *Enhanced Interior Gateway Routing Protocol, White Paper Document ID: 16406*, from http://www.cisco.com/warp/public/103/eigrp-toc.html

Cisco Systems 2005b, *Interior Gateway Routing Protocol, Internetworking Technology Handbook*, from http://www.cisco.com/univercd/cc/td/doc/cisintwk/ito_doc/igrp.htm

Clausen, T. & Jacquet, P. 2004, *The Optimised Routing Protocol for Mobile Ad-hoc Networks: protocol specification*, Institut National de Recherche en Informatique et en Automatique (INRIA).

Coltun, R., Ferguson, D. & Moy, J. 1999, 'RFC2740: OSPF for IPv6', *Internet Engineering Task Force, Network Working Group*.

Deering, S. & Hinden, R. 1998, 'RFC2460: Internet Protocol Version 6 (IPv6) Specification', *Internet Engineering Task Force, Network Working Group*.

Diestel, R. 2000, *Graduate Texts in Mathematics - Graph Theory,* 2nd edn, Springer-Verlag, Heidelberg.

Dijkstra, E.W. 1959, 'A Note on Two Problems in Connexion with Graphs', *Numerische Mathematik,* vol. 1, pp. 269-71.

Du, L. & Chen, L. 2005, 'Receiver Initiated Network Allocation Vector Clearing Method in WLANs', *The 2005 Asia-Pacific Conference on Communications*, Perth, Australia, pp. 615-9.

Ehsan, H. & Uzmi, Z.A. 2004, 'Performance comparison of ad hoc wireless network routing protocols', *Multitopic Conference, 2004. Proceedings of INMIC 2004. 8th International*, pp. 457-65.

ETSI GSM 1997, *Digital Cellular Telecommunications System (Phase 2+); High Speed Circuit Switched Data (HSCSD) - Stage 1 (GSM 02.34)*, European Telecommunications Standards Institute.

ETSI GSM 1998, *Digital Cellular Telecommunications System (Phase 2+); General Packet Radio Service (GPRS); Service Description; Stage 1 (GSM 02.60)*, European Telecommunications Standards Institute.

Ferrari, G. & Tonguz, O.K. 2007, 'Impact of Mobility on the BER Performance of Ad Hoc Wireless Networks', *Vehicular Technology, IEEE Transactions on,* vol. 56, no. 1, pp. 271-86.

Fourty, N., Val, T., Fraisse, P. & Mercier, J.J. 2005, 'Comparative analysis of new high data rate wireless communication technologies "From Wi-Fi to WiMAX"', *Joint International Conference on Autonomic and Autonomous Systems (ICAS) and International Conference on Networking and Services (ICNS) 2005*, pp. 66-.

Fu, Z., Zerfos, P., Luo, H., Lu, S., Zhang, L. & Gerla, M. 2003, 'The impact of multihop wireless channel on TCP throughput and loss', *22nd Annual Joint Conference of the IEEE Computer and Communications Societies,* vol. 3, pp. 1744-53.

Garcia-Luna-Aceves, J.J. 1993, 'Loop-free routing using diffusing computations', *IEEE/ACM Transactions on Networking,* vol. 1, no. 1, pp. 130-41.

Garcia-Luna-Aceves, J.J. & Spohn, M. 1999, 'Source-tree routing in wireless networks', *Network Protocols, 1999. (ICNP '99) Proceedings. Seventh International Conference on*, pp. 273-82.

Giannoulis, S., Katsanos, C., Koubias, S. & Papadopoulos, G. 2004, 'A hybrid adaptive routing protocol for ad hoc wireless networks', *Factory Communication Systems, 2004. Proceedings. 2004 IEEE International Workshop on*, pp. 287-90.

Goldsmith, A. 2005, *Wireless Communications*, Cambridge University Press, New York.

Gouda, M.G. & Schneider, M. 2003, 'Maximizable routing metrics', *IEEE/ACM Transactions on Networking,* vol. 11, no. 4, pp. 663-75.

Guo, S., Yang, O. & Shu, Y. 2005, 'Improving source routing reliability in mobile ad hoc networks', *Parallel and Distributed Systems, IEEE Transactions on,* vol. 16, no. 4, pp. 362-73.

Guo, S. & Yang, O.W. 2002, 'Performance of backup source routing in mobile ad hoc networks', *Wireless Communications and Networking Conference, 2002. WCNC2002. 2002 IEEE*, pp. 440-4 vol.1.

Haas, Z.J. 1997, 'A new routing protocol for the reconfigurable wireless networks', *Universal Personal Communications Record, 1997. Conference Record., 1997 IEEE 6th International Conference on*, pp. 562-6 vol.2.

Haas, Z.J. & Pearlman, M.R. 1998a, 'The performance of query control schemes for the zone routing protocol', in *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication*, ACM Press, Vancouver, British Columbia, Canada, pp. 167-77.

Haas, Z.J. & Pearlman, M.R. 1998b, 'The Zone Routing Protocol (ZRP) for Ad Hoc Networks', *IETF MANET Working Group, Internet Draft, draft-ietf-manet-zone-zrp-01.txt*.

Haas, Z.J. & Pearlman, M.R. 2001, 'The performance of query control schemes for the zone routing protocol', *Networking, IEEE/ACM Transactions on,* vol. 9, no. 4, pp. 427-38.

Hao, Z., Wei, X., Xiong, N. & He, Y. 2009, 'Routing in 802.11-Based Wireless Mesh Networks', *Computational Science and Engineering, 2009. International Conference on,* vol. 2, pp. 1007-12.

Hawkinson, J. & Bates, T. 1996, 'RFC1930 / BCP6: Guidelines for creation, selection, and registration of an Autonomous System (AS)', *Internet Engineering Task Force, Network Working Group*.

Hedrick, C. 1988, 'RFC1058: Routing Information Protocol', *Internet Engineering Task Force, Network Working Group*.

Hiertz, G.R., Denteneer, D., Max, S., Taori, R., Cardona, J., Berlemann, L. & Walke, B. 2010, 'IEEE 802.11s: The WLAN Mesh Standard', *Wireless Communications, IEEE,* vol. 17, no. 1, pp. 1536-284.

Hu, Y.-C. & Johnson, D.B. 2000, 'Caching strategies in on-demand routing protocols for wireless networks', *ACM/IEEE MobiCom*, pp. 231-42.

Hu, Y.-C. & Johnson, D.B. 2002, 'Ensuring cache freshness in on-demand ad hoc network routing protocols', in *Proceedings of the second ACM international workshop on Principles of mobile computing*, ACM Press, Toulouse, France, pp. 25-30.

Hunt, C. 1997, *TCP/IP Network Administration,* 2nd edn, O'Reilly & Associates.

IANA 2008, *IPv4 Address Report*, from http://www.potaroo.net/tools/ipv4/index.html

IEEE Draft Std 802.11s 2010, *Status of Project IEEE 802.11s, Meetings Update*, from http://grouper.ieee.org/groups/802/11/Reports/tgs_update.htm

IEEE Std 802.3 2005, *Part 3 - Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications*, The Institute of Electrical and Electronics Engineers, Inc., New York, USA.

IEEE Std 802.11 1999, *Part 11 - Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specification*, The Institute of Electrical and Electronics Engineers, Inc., New York, USA.

IEEE Std 802.11n 2009, *Amendment 5: Enhancements for Higher Throughput*, The Institute of Electrical and Electronics Engineers, Inc., New York, USA.

IEEE Std 802.16 2004, *Part 16 - Air Interface for Fixed Broadband Wireless Access Systems*, The Institute of Electrical and Electronics Engineers, Inc., New York, USA.

ITU 1999, *Recommendation Q.1701: Framework for IMT-2000 networks*, International Telecommunications Union, Geneva, Switzerland.

ITU 2004, *Recommendation M.1677: International Morse code*, International Telecommunications Union, Geneva, Switzerland.

Jensen, P.A. & Bard, J.F. 2002, *Operations Research Models and Methods*, John Wiley and Sons, New York.

Joa-Ng, M. & Lu, I.-T. 1999, 'A peer-to-peer zone-based two-level link state routing for mobile ad hoc networks', *Selected Areas in Communications, IEEE Journal on,* vol. 17, no. 8, pp. 1415-25.

Johnson, D.B. & Maltz, D.A. 1996, 'Dynamic source routing in ad hoc wireless networks', *Mobile Computing, IEEE Transactions on,* vol. 353.

Johnson, D.S. 1973, 'Approximation algorithms for combinatorial problems', in *Proceedings of the fifth annual ACM symposium on Theory of computing*, ACM Press, Austin, Texas, United States, pp. 38-49.

Karp, R.M. 1972, 'Reducibility Among Combinatorial Problems', in Miller, R.E. & Thatcher, J.W. (eds), *Complexity of Computer Computations*, Plenum Press, New York, pp. 85-103.

Kirchhoff, G. 1847, 'Uber die Auflosung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Strome gefuhrt wird', *Ann. Phys. Chem.,* vol. 72, pp. 497-508.

Koutsonikolas, D., Das, S.M., Pucha, H. & Hu, Y.C. 2005, 'On optimal TTL sequence-based route discovery in MANETs', *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pp. 923-9.

Kozierok, C.M. 2005, *The TCP/IP Guide: A Comprehensive, Illustrated Internet Protocols Reference*, No Starch Press, Inc.

Lee, S.-J., Belding-Royer, E.M. & Perkins, C.E. 2003, 'Scalability study of the ad hoc on-demand distance vector routing protocol', *Int. J. Netw. Manag.,* vol. 13, no. 2, pp. 97-114.

Leung, R., Liu, J., Poon, E., Chan, A.-L.C. & Li, B. 2001, 'MP-DSR: a QoS-aware multi-path dynamic source routing protocol for wireless ad-hoc networks', *Local Computer Networks, 2001. Proceedings. LCN 2001. 26th Annual IEEE Conference on*, pp. 132-41.

Li, J., Blake, C., De Couto, D.S.J., Lee, H.I. & Morris, R. 2001, 'Capacity of Ad Hoc wireless networks', in *Proceedings of the 7th annual international conference on Mobile computing and networking*, ACM Press, Rome, Italy, pp. 61-9.

Li, X., Kong, P.-Y. & Chua, K.-C. 2007, 'TCP Performance in IEEE 802.11-Based Ad Hoc Networks with Multiple Wireless Lossy Links', *Mobile Computing, IEEE Transactions on,* vol. 6, no. 12, pp. 1329-42.

Lopez, J., Barcelo, J.M. & Garcia-Vidal, J. 2005, 'Analysing the overhead in Mobile ad-hoc network with a hierarchical routing structure', *International Working Conference Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETS) 2005*.

Lougheed, K. 1990, 'RFC1163: A Border Gateway Protocol (BGP)', *Internet Engineering Task Force, Network Working Group*.

Lu, Y., Wang, W., Zhong, Y. & Bhargava, B. 2003, 'Study of distance vector routing protocols for mobile ad hoc networks', *Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on*, pp. 187-94.

Maltz, D.A., Broch, J., Jetcheva, J. & Johnson, D.B. 1999, 'The effects of on-demand behavior in routing protocols for multihop wireless ad hoc networks', *Selected Areas in Communications, IEEE Journal on,* vol. 17, no. 8, pp. 1439-53.

Manshaei, M.H., Cantieni, G.R., Barakat, C. & Turletti, T. 2005, 'Performance Analysis of the IEEE 802.11 MAC and Physical Layer Protocol', in *Proceedings of the Sixth IEEE International Symposium on World of Wireless Mobile and Multimedia Networks*. IEEE Computer Society.

Marina, M.K. & Das, S.R. 2001, 'Performance of route caching strategies in Dynamic Source Routing', *Distributed Computing Systems Workshop, 2001 International Conference on*, pp. 425-32.

Mills, D.L. 1984, 'RFC904: Exterior Gateway Protocol Formal Specification', *Internet Engineering Task Force, Network Working Group*.

Miranda, H. & Rodrigues, L. 2005, 'Using a fairness monitoring service to improve load-balancing in DSR', *Distributed Computing Systems Workshops, 2005. 25th IEEE International Conference on*, pp. 314-20.

Moy, J. 1989, 'RFC1131: Open Shortest Path First (OSPF)', *Internet Engineering Task Force, Network Working Group*.

Moy, J. 1998, 'RFC2328 / STD 54: OSPF Version 2', *Internet Engineering Task Force, Network Working Group*.

Ni, S.-Y., Tseng, Y.-C., Chen, Y.-S. & Sheu, J.-P. 1999, 'The broadcast storm problem in a mobile ad hoc network', in *Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, ACM Press, Seattle, Washington, United States, pp. 151-62.

Nikaein, N., Labiod, H. & Bonnet, C. 2000, 'DDR-distributed dynamic routing algorithm for mobile ad hoc networks', *Mobile and Ad Hoc Networking and Computing, 2000. MobiHOC. 2000 First Annual Workshop on*, pp. 19-27.

NS-2 2005, *The Network Simulator - ns-2*, from http://www.isi.edu/nsnam/ns

Pearlman, M.R. & Haas, Z.J. 1999, 'Determining the optimal configuration for the zone routing protocol', *Selected Areas in Communications, IEEE Journal on,* vol. 17, no. 8, pp. 1395-414.

Perkins, C.E. 1997, 'Ad Hoc On Demand Distance Vector (AODV) Routing', *IETF MANET Working Group, Internet Draft, draft-ietf-manet-aodv-00.txt*.

Perkins, C.E. & Bhagwat, P. 1994, 'Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers', in *Proceedings of the conference on Communications architectures, protocols and applications*, ACM Press, London, United Kingdom, pp. 234-44.

Perkins, C.E., Royer, E.M., Das, S.R. & Marina, M.K. 2001, 'Performance comparison of two on-demand routing protocols for ad hoc networks', *Personal Communications, IEEE [see also IEEE Wireless Communications],* vol. 8, no. 1, pp. 16-28.

Plummer, D. 1982, 'RFC 8126: An Ethernet address resolution protocol: or converting network protocol addresses to 48-bit Ethernet address for transmission on Ethernet hardware', *Internet Engineering Task Force*.

Postel, J. 1981, 'RFC791: DARPA Internet Program Protocol Specification', *Defense Advanced Research Projects Agency (DARPA)*.

Preiss, B.R. 1998, *Data Structures and Algorithms with Object-Oriented Design Patterns in Java*, University of Waterloo, Waterloo, Canada, from http://www.brpreiss.com/books/opus5/html/book.html

Proakis, J.G. & Salehi, M. 2001, *Communication Systems Engineering,* 2nd edn, Prentice-Hall.

Rappaport, T. 2002, *Wireless Communications Principles and Practice,* 2nd edn, Prentice-Hall, Inc., New Jersey, USA.

Rekhter, Y., Li, T. & Hares, S. 2006, 'A Border Gateway Protocol 4 (BGP-4)', *Internet Engineering Task Force, Network Working Group*.

Salus, P.H. 1995, *Casting the Net from ARPAnet to Internet and Beyond*, Addison-Wesley Publishing Company.

Samar, P., Pearlman, M.R. & Haas, Z.J. 2004, 'Independent zone routing: an adaptive hybrid routing framework for ad hoc wireless networks', *Networking, IEEE/ACM Transactions on,* vol. 12, no. 4, pp. 595-608.

Shea, K., Ives, R.W. & Tummala, M. 2000, 'Mobile ad hoc network routing protocol analysis and its application to a programmable modular communication system', *Signals, Systems and Computers, 2000. Conference Record of the Thirty-Fourth Asilomar Conference on*, pp. 1260-4 vol.2.

Shuaib, A.H. & Aghvami, A.H. 2009, 'A Routing Scheme for the IEEE-802.15.4-Enabled Wireless Sensor Networks', *Vehicular Technology, IEEE Transactions on* vol. 58, no. 9, pp. 5135-51.

Smith, C. & Collins, D. 2002, *3G Wireless Networks*, McGraw-Hill, Singapore.

Thomas, T.M. 2003, *OSPF Network Design Solutions,* 2nd edn, Cisco Press.

TIA 1999, *Mobile Station-Base Station Compatibility Standard for Wideband Spread Spectrum Cellular Systems (TIA/EIA-95-B)*, Telecommunications Industry Association.

Toh, C.-K., Le, A.-N. & Cho, Y.-Z. 2009, 'Load balanced routing protocols for ad hoc mobile wireless networks', *Communications Magazine, IEEE,* vol. 47, no. 8, pp. 78-84.

Valera, A.C., Seah, W.K.G. & Rao, S.V. 2005, 'Improving protocol robustness in ad hoc networks through cooperative packet caching and shortest multipath routing', *Mobile Computing, IEEE Transactions on,* vol. 4, no. 5, pp. 443-57.

Van Der Werf, S.M. & Chung, K.-S. 2004, 'Multihop Wireless Ad-hoc Routing Protocol (MultiWARP)', *The 2004 Joint Conference of the 10th Asia-Pacific Conference on Communications, and The 5th International Symposium on Multi-Dimensional Mobile Communications*, Beijing, China, pp. 961-5 vol.2.

Van Der Werf, S.M. & Chung, K.-S. 2005, 'Performance of MultiWARP Routing Protocol for Multi-hop Wireless Ad-hoc Network', *The 2005 Asia-Pacific Conference on Communications*, Perth, Australia, pp. 779-83.

Wang, L. & Olariu, S. 2004, 'A two-zone hybrid routing protocol for mobile ad hoc networks', *Parallel and Distributed Systems, IEEE Transactions on,* vol. 15, no. 12, pp. 1105-16.

Wang, L., Shu, Y., Dong, M., Zhang, L. & Yang, O.W.W. 2001, 'Adaptive multipath source routing in ad hoc networks', *Communications, 2001. ICC 2001. IEEE International Conference on*, pp. 867-71 vol.3.

Wu, D.-Y., Hou, Z.-F. & Hou, C.-Z. 2003, 'Improved caching strategies in on-demand routing protocols for mobile ad hoc networks', *Communication Technology Proceedings, 2003. ICCT 2003. International Conference on*, pp. 1258-61 vol.2.

Wu, J. 2002, 'An extended dynamic source routing scheme in ad hoc wireless networks', *System Sciences, 2002. HICSS. Proceedings of the 35th Annual Hawaii International Conference on*, pp. 3832-8.

Wu, S.-L., Ni, S.-Y., Tseng, Y.-C. & Sheu, J.-P. 2000, 'Route maintenance in a wireless mobile ad hoc network', *System Sciences, 2000. Proceedings of the 33rd Annual Hawaii International Conference on*, p. 10 pp. vol.2.

Xu, S. & Saadawi, T. 2001, 'Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks?', *Communications Magazine, IEEE,* vol. 39, no. 6, pp. 130-7.

Yang, C.-C. & Tseng, L.-P. 2005, 'Fisheye zone routing protocol for mobile ad hoc networks', *Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE*, pp. 1-6.

Yoon, J., Liu, M. & Noble, B. 2003, 'Random Waypoint Considered Harmful', *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies. IEEE,* vol. 2, pp. 1312-21.

Zhang, X. & Jacob, L. 2003a, 'Adapting zone routing protocol for heterogeneous scenarios in ad hoc networks', *Parallel Processing, 2003. Proceedings. 2003 International Conference on*, pp. 341-8.

Zhang, X.F. & Jacob, L. 2003b, 'Multicast Zone Routing Protocol in Mobile Ad Hoc Wireless Networks ', in *Proceedings of the 28th Annual IEEE International Conference on Local Computer Networks* IEEE Computer Society, p. 150

Zhang, X.M., Zou, F.F., Wang, E.B. & Sung, D.K. 2010, 'Exploring the Dynamic Nature of Mobile Nodes for Predicting Route Lifetime in Mobile Ad Hoc Networks', *Vehicular Technology, IEEE Transactions on,* vol. 59, no. 3, pp. 1567-72.

Zhao, S., Wu, Z., Acharya, A. & Raychaudhuri, D. 2005, 'PARMA: a PHY/MAC aware routing metric for ad-hoc wireless networks with multi-rate radios', *World of Wireless Mobile and Multimedia Networks, 2005. WoWMoM 2005. Sixth IEEE International Symposium on a*, pp. 286-92.

Zimmermann, H. 1980, 'OSI Reference Model - The ISO Model of Architecture for Open Systems Interconnection', *IEEE Transactions on Communications,* vol. 28, no. 4, pp. 425-32.