MANET Routing with Prediction

Raymee Chau

Thesis submitted for the degree of

Master of Engineering Science

in

Electrical and Electronic Engineering

at

The University of Adelaide

School of Electrical and Electronic Engineering

Faculty of Engineering, Computer & Mathematical Sciences

November, 2012

Contents

Abstract

Route stability of Mobile Ad-Hoc Networks (MANETs) is one of the major problems in defence tactical wireless networks. The dynamic nature of MANETs may cause the network topology to change frequently as a result of unstable links, which may result in frequent route changes. Unstable routes may cause retransmissions and drop outs. Therefore, the network can experience heavy traffic overload and high packet losses. Many network applications rely on a stable and reliable route. Hence, it is important for the military to have a reliable network that allows effective communications amongst various platforms to effectively perform the tasks they have been assigned. For this reason, the route's stability in MANETs needs to be understood. However, many existing MANET routing protocols are not explicitly designed for route stability. It is expected that prediction can assist in increasing a MANET's route stability. This thesis explores the potential benefits and the trade-offs in the use of prediction with the Ad-hoc On-demand Distance Vector (AODV) routing protocol.

In the context of using prediction in routing, research has shown that using "accurate" predictions can improve MANETs' routing performance. However, Chapter 3 shows that it is difficult to achieve accurate predictions. To the author's knowledge, very little work has been attempted to analyse the routing performance with reduced prediction accuracies, and the effects of having inaccurate prediction. Thus more specifically, this thesis examines the robustness of using link duration prediction with various accuracies for MANETs, and identifies the conditions for which predictions can improve routing performance.

This is achieved by first examining how using perfectly accurate link duration prediction can improve routing performance. For this purpose, a new routing protocol, Ad-hoc On-demand Distance Vector with Perfect Prediction (AODV-PP), has been created to propagate link duration prediction information for route establishment. The OPNET simulator was used to simulate network scenarios with AODV and AODV-PP for analysis, and the routing performance of the two protocols have been compared.

The thesis later explores how inaccurate link duration prediction affects routing performance. However, the AODV-PP protocol does not inform the source about the change in predicted link duration. This can cause delays in route re-establishment and high packet loss. Hence, AODV with Prediction Update (AODV-PU) has been proposed to allow link duration prediction updates to be sent to the source for route maintenance. Network scenarios with AODV-PU were simulated to analyse and compare its routing performance with AODV and AODV-PP.

This thesis shows stable routes can be found with perfect prediction, which reduces packet loss and routing overhead. However, it also indicates that it is difficult to use link duration prediction to find a more stable route with inaccurate long-term predictions. Nevertheless, link duration prediction can be useful for route updates and route re-establishments, which only requires short-term predictions, to allow more seamless route transitions and to reduce packet loss. The trade-off being that more control traffic is required for route maintenance. This in turn creates a more robust platform for the military applications that require this type of network.

Statement of Originality

Acknowledgment

This thesis is made possible with the assistance of many people. I would like to express my deep appreciation to all those who have provided me support and guidance.

Firstly, I would like to express my sincere gratitude to my supervisors, Professor Langford White, Dr Andrew Coyle and Dr Michael Rumsewicz, at the University of Adelaide, for their patience, advise, support and guidance. Their expertise and insightful suggestions have clarified my ideas, provided advise in analysing and presenting the results, and expanded my visions in other related research areas. In particular, Dr Andrew Coyle has provided many valuable comments and suggestions towards this thesis. And I would like to thank those from the Centre for Defence Communications and Information Network at the University of Adelaide who have assisted.

I would also like to thank my managers at the Defence Science and Technology Organisation (DSTO), Dr Peter Shoubridge, Dr Jeff McCarthy, Dr Perry Blackmore and Mr Darren Wilksch for the support, opportunity and flexibility provided for me to do this study. Finally, I wish to thank my colleagues at DSTO who have provided me advice and guidance on technical issues, results analysis and thesis writing.

Acronyms

AODV	Ad-hoc On-demand Distance Vector
AODV-PP	AODV with Perfect Prediction
AODV-PU	AODV with Prediction Update
ARP	Address Resolution Protocol
BATMAN	Better Approach To Mobile Adhoc Networking
CTS	Clear-to-Send
CVBP	Constant-Velocity Based Prediction
DSDV	Destination Sequenced Distance Vector
DSR	Dynamic Source Routing
EMM	Entity Mobility Model
FSR	Fisheye State Routing
GMM	Gauss Markov Model
GPS	Global Positioning Systems
HMM	Hidden Markov Model
IARP	Intrazone Routing Protocol
IPLC	Incorrectly Predicted Link Change
IPLS	Incorrectly Predicted Link State
LAN	Local Area Network
LET	Link Expiration Time
MAC	Medium Access Control
MANET	Mobile Ad-hoc Network
OLSR	Optimized Link State Routing

OPNET	Optimized Network Engineering Tools
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
OSPF-MDR	OSPF MANET Designated Routers (extension of OSPFv3 to support MANETs)
PreRREQ	Pre-Route RREQ
RERR	Route Error
RMSE	Root Mean Square Error
RREP	Route Reply
RREQ	Route Request
RSS	Received Signal Strength
RTS	Request-to-Send
RUPDATE	Route Update
RWM	Random Walk Model
RWP	Random Waypoint
SNR	Signal-to-Noise Ratio
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TTL	Time-To-Live
UDP	User Datagram Protocol
ZRP	Zone Routing Protocol

Chapter 1

Introduction

A Mobile Ad-Hoc Network (MANET) is a dynamic multihop infrastructureless self-forming mobile network. In a MANET all the nodes are able to operate as a router as well as a host, so that any node has the ability to send, receive and route traffic within the same network. Such infrastructureless networks are useful in the military, where forces may be deployed at a location where there is no reliable communication infrastructure; or for disaster relief, where the fixed network infrastructure has been disrupted. Also in a highly dynamic environment using fast moving platforms, the structure of any network is changing so frequently that the routing protocols designed for fixed topology networks may not be capable of handling the network's information requirements. For these and other scenarios it is important to have a reliable MANET[1] to enable communication to occur in the network efficiently and robustly.

Routing performance[2] of MANETs [1] has been a major area of research for a number of years. In the last decade, research has shown that accurate mobility or link state predictions can improve the routing performance of MANETs

[1]A reliable MANET means having reliable end-to-end connectivity between communicating nodes with low jitter, end-to-end delay, packet loss and loss of connectivity.

[2]Routing performance includes jitter, end-to-end delay, packet loss, routing overhead traffic, and route setup time.

[2–15]. The benefits of accurate network state prediction include the following: reducing the route re-convergence times; reducing the packet loss and delay; and increasing the stability and reliability of the routes. However, very little work has been done on comparing the robustness of proposed prediction methods and subsequent effects on routing performance. It is important to understand the impacts of the prediction accuracy on measures of routing performance. This research project examines the robustness of existing prediction methods proposed for MANETs and identifies the conditions for which predictions can improve routing performance. This is undertaken by modifying the existing Ad-hoc On-Demand Distance Vector (AODV) [16] routing protocol so that a realistic study of these effects may be undertaken. AODV is chosen for the modification is because AODV is a commonly used routing protocol for research, which is readily available in various simulation tools, and it is easily modifiable.

1.1 Research Motivation

The nodes in a MANET are mobile and are connected wirelessly. Between any two nodes, the wireless connection may constantly change because of the changing attenuation due to distance, and different fading and interference patterns. Consequently, the connectivity between nodes may change. The building blocks of any route through the MANET are these unstable links, and thus the routes through the network will change with the changing network conditions. As a result, the routes may become less reliable, which affects the routing performance of MANETs in the following ways:

1. Frequent route changes;
2. Fluctuation in end-to-end delays;
3. Varying packet loss rates;
4. Out of order packet arrival;
5. Changing amounts of routing overhead traffic; and
6. Varying traffic loads on the network due to retransmission of data traffic.

Over the years, many attempts have been made to improve the routing performance of MANETs. A typical attempt to enhance the performance of a MANET is to develop new and improved routing protocols. To date, many routing protocols have been proposed in the literature, yet there is no single routing protocol that is well suited for all environments in which MANETs may operate [17–24]. This is because of the dynamic nature of MANETs where nodes can move freely, the network density varies, the network size changes as nodes join and leave the network, the wireless channel characteristics fluctuates, and traffic patterns fluctuate. The protocols that are robust for fast moving nodes may not be suitable for slow moving nodes, and the ones that are suitable for large networks may not work well for small networks. Also the protocols that are more appropriate for highly loaded networks may not be appropriate for networks with light traffic.

In many existing routing methods, routes are formed and maintained based on the currently received information of the network. A single snap shot of the network is used to determine the best routes through the network. Such information may be outdated at the time the information is being received and used, especially for long duration information transfers. The change of link state, from active to in-active, may not be known to the sender at the time the node sends a packet. This may result in packet loss and/or increased delay in delivering the packet to the destination. Moreover, the network information obtained may not be accurate, which causes difficulties in establishing and maintaining the routes. This is because most of the existing routing protocols (for both reactive and proactive protocols) are reactive to the change in link availability, i.e. the protocol waits until a link problem is detected before it searches for a new route. Thus, the routes may be unstable depending on how the routes are selected. Furthermore, traditional routing can only be accomplished if all the links along a path are connected at any one time. Therefore, it may be difficult to route packets successfully through a dynamic MANET. This is more so for a sparse MANET, where the connectivity of the network at any one time may be low, thus sending packets to a currently isolated node may not be possible. This is why many of the existing routing methods may not always be efficient and effective.

Some routing strategies attempt to mitigate these effects by looking at average link packet loss. A protocol such as Better Approach To Mobile Adhoc Networking (BATMAN) [25] uses the historical record of link packet loss to better choose a route through the network. But this approach works mainly in a fixed network in which the past behavior is a reliable guide to future performance. Where the nodes are mobile and so the links connectivity is changing in time this type of approach no longer works. In fact by using some out of date historical information these protocols may hold onto a route longer than is necessary after a radio link change has occurred. (As opposed to a fluctuation link[3] for which these strategies are primarily aimed.)

Route stability is one of the most crucial issues in MANET routing, and this is more so for the military networks. For example, in situations where there are fast moving platforms moving in and out of range of the slow moving platforms, low stability links may form between the slow and the fast moving platforms. During route discovery, the routing protocol then needs to make a decision between the route through a shorter but less stable route or a longer but more stable one. The choice made would affect the routing performance differently depending on the environment and the situation. Frequent route discoveries due to choosing less stable links can result in an increase in routing overhead traffic, increase packet loss and the order of packet arrival. This can increase traffic loads on the network, and can cause changes in end-to-end delays. Some applications are not robust to these varying route conditions and will not work in these conditions. Hence by increasing the stability of the route, routing issues such as packet loss, routing overhead traffic and data traffic loads can be reduced. This in turn enables a better performance of the MANET for the user. Thus, a more effective way of routing in a dynamic environment may be to predict changes of the network state and prepare for the changes in advance. Whereas standard MANET protocols do not take the expected longevity of a link into consideration when choosing a route through a network, one based on prediction can filter out expected short lived links and routes. This leaves the longer lived, more robust, routes which in turn increases the performance of the network for the users.

[3] A fluctuation link is a link that is affected by fading and interference.

Routing through platforms with multi-bearers is another area where prediction may be useful. For instance, when multi-bearers (with a wide range of frequencies) are available on each of the platforms, some bearers (usually with lower frequencies) may form more stable links compared to other bearers (that are in the higher frequencies), however, low frequency channels will often have a lower data rate compared to high frequency channels. Furthermore, different frequencies may be more reliable in different weather conditions and different environment. This may further complicate the selection of links for routing. If a standard MANET routing protocol is used, it is expected to choose the shortest path or the one with the least cost, however this may not be the ideal choice. For example, consider a network which has a satellite link connected to all the nodes in the network, so there is always a one hop route (through the satellite) between all the nodes. This satellite route is then likely to be chosen for all routes. However, the satellite link may be unstable due to adverse weather conditions. In this situation, other non-satellite link routes should be used. The longevity of the unstable satellite links should inform the routing protocol, and this information used accordingly.

Various strategies have been proposed in an attempt to improve route stability. One strategy is to select a route that satisfies different conditions that would affect the stability of the route. Koul *et al.* [26] have used the current parameters such as distance between two nodes, frequency and Receiver Signal Strength Indicator (RSSI) to decide whether a link should be used. However, this method only uses the current parameters, no historical records nor future predictions have been used in the route selection decision making.

By using network state prediction, routes can be constructed based on predicted link duration, which can be based on the mobility of the nodes and/or the stability of the link, rather than relying on the currently observed network state information that may be outdated. Using prediction may reduce the likelihood of using an unreliable or a fully loaded link, and may also reduce the need for re-routing by maximising the lifetime of a route. In turn, this may reduce packet loss, end-to-end delay and routing overhead traffic. Furthermore, if one can predict that a node will be connected to the destination at some point in time, packets may be forwarded to that node even though there is no physical route to the final destination, and those packets can be kept in that node for a period of time or until a route to the final destination becomes available.

On the other hand, it is uncertain whether or not routing with prediction is realistic. In reality, information may be inaccurate, lost or outdated, and predictions may not always be accurate. Thus, if prediction is to be used in routing, it is important to explore the maximum benefits and the trade-offs of using prediction, the degree of prediction accuracy required, and the feasibility of using prediction for MANET routing. Furthermore it is also important to address the feasibility and the issues of implementing a routing protocol using prediction, and whether current well established MANET routing protocols can be modified simply so that the predictive effects will be added to the current capabilities of this protocol.

1.2 Objectives

The objectives of this research study are to:

1. Investigate whether mobility prediction is feasible in the context of MANET routing;
2. Learn what degree of accuracy is required for making practical improvements in routing performance;
3. Establish whether prediction accuracy can be improved to an extent where it will be useful for routing; and
4. Determine whether an existing MANET protocol can be realistically modified to include prediction.

These objectives can be achieved by doing the following:

1. Determining an appropriate way to evaluate the accuracy of mobility prediction methods for MANET routing;
2. Identifying, analysing and exploring the factors that affect the accuracy of

existing prediction methods;

3. Evaluating the effects of prediction accuracy on network routing performance, in order to determine the utility of using prediction in routing for dynamic environments; and
4. Modifying the AODV routing protocol to introduce prediction, simulating this enhanced AODV protocol, and analysing the performance of routing with mobility prediction using the resultant simulation.

1.3 Outline

Chapter 2 provides some background knowledge to this thesis. This includes information on various types of MANET routing protocols in the literature, the mobility models that are commonly used by MANET researchers, and the prediction methods used.

Chapter 3 studies three different evaluation metrics that can be used for evaluating the accuracy of prediction for MANETs. Two of the three evaluation metrics are chosen to evaluate the accuracy of three different mobility prediction methods. The findings of this study are summarised at the end of this chapter.

Chapter 4 compares the routing performance between AODV routing protocol and AODV with perfect prediction. This study provides a benchmark for the best performance AODV can possibly achieve when prediction is used.

Chapter 5 looks at the effect of inaccurate prediction, and how prediction accuracy affects routing performance. In this chapter, the routing performance is evaluated for cases where prediction is not always accurate.

Chapter 6 summarises the accomplishments and contributions of the study, and discuss some future directions of the research.

Chapter 2

Background and Literature Review

2.1 Introduction

This chapter provides an overview of the literature that is related to MANET routing using predictions and prediction methods that have been used for wireless networks in general. Section 2.2 explains different types of routing protocols. Section 2.3 provides an introduction to some common mobility models that have been used for MANET simulations. The various types of prediction methods that have been studied or used for MANET routing are explained in Section 2.4.

2.2 Types of MANET Routing Protocols

Routing protocols are required in a network to search for and maintain routes between a traffic source and the destination of that traffic. Although routing protocols such as the traditional Open Shortest Path First (OSPF) [27] have been widely used on wired networks for decades, and have worked effectively, they are not sufficient for dynamic networks with constantly changing topologies and unstable link connectivity. These protocols can only react to changes in the network, as they are designed for networks with a static topology. When there is a topology change, OSPF needs to propagate a large amount of overhead traffic, which is not appropriate for MANETs which often have limited resources.

In military networks, it is crucial to have reliable infrastructureless mobile communications to support tactical military operations, hence a reliable MANET routing protocol is necessary. Whereas a fixed infrastructure is more reliable in urban environments, the military often operate in isolated localities. Whether in a military or a humanitarian operation, what fixed infrastructure had existed may no longer be in operation, networks must be quickly deployed and are likely to be dynamic. In spite of the known issues with MANETs in these circumstances, they are usually the only option available, and to be effective, they must be made as robust as possible.

For these and other reasons, over the last couple of decades many routing protocols have been proposed and evaluated. However, not many have been successfully deployed on a real MANET, as many of these routing protocols do not work well in practice. Many of the proposed MANET routing protocols can only react to changes in the network after the change has occurred. A route may be set up after a link has been broken for a short period of time. During this time, not only has the route not carried the traffic which requested the route, but possibly limited resources have been expended on overheads necessary for the setting up of the route.

In general, MANET routing protocols can be classified into the following categories:

1. Proactive (table-driven) versus reactive (on-demand);
2. Distance vector versus link state;
3. Flat versus Hierarchical; and
4. With Prediction versus without Prediction.

2.2.1 Proactive Routing Protocols

Proactive routing protocols such as Destination Sequenced Distance Vector (DSDV) [28], Optimized Link State Routing (OLSR) [29] and OLSR version 2 (OLSRv2) [30] which is a successor of OLSR, actively search for new routes and maintain existing routes periodically, regardless of whether the routes are being used or not. All possible routes through the network are calculated using efficient shortest path algorithms. When using proactive routing protocols, every node in the network updates its routing information at a regular interval, to maintain the knowledge of the route or the next hop to any reachable destination. Since the path to the destination is known, there is no need for route discovery when there is a packet to be sent. This may reduce the path initialisation latency and increase the reliability of the network. However, constant routing overhead traffic is required to maintain the routing table, which consumes a lot of bandwidth.

Therefore with proactive routing protocols, it is important to choose an appropriate updating interval for different rate of change of network topology, as a lack of route updates may result in invalid routes and frequent route updates may result in high overhead. For example, a highly dynamic network may need more frequent route updates in respect to a slow moving network, but on the other hand, the more frequent the route update is, the more routing traffic will be sent, and hence requires higher bandwidth consumption. This may be an issue if the bandwidth is limited.

In [31], Yadav and Mishra have compared the performance between four proactive routing protocols: Fisheye State Routing (FSR), OLSR, Intrazone Routing Protocol (IARP) and Bellman-Ford Routing Protocol. This study shows that OLSR performs better in general, but IARP seems slightly better in terms of end-to-end delay, while FSR performs very poorly. This study indicates that the performance of different proactive routing protocols can vary greatly.

2.2.2 Reactive Routing Protocols

In reactive routing protocols like Dynamic Source Routing (DSR) [32], Ad-hoc On-demand Distance Vector (AODV) [16] and Dynamic MANET On-demand (DYMO or AODVv2) [33] which is a successor of AODV, routes are only requested and maintained when there are packets to be sent. The main advantage of such routing protocols over proactive routing protocols is that it reduces the routing overhead traffic, with a trade-off of having longer path initialisation times. This is especially true if a small proportion of the available routes are used. Reactive routing protocols are generally suitable for more dynamic environment or networks with high mobility.

A number of comparison studies have been made between proactive and reactive routing protocols [34–37]. The studies in [34, 35] compared the AODV, DSR and DSDV protocols has shown that DSR performs better than AODV and DSDV, while DSDV performs more poorly comparatively. Whereas the studies in [36, 37] have made a comparison between AODV, DSR and OLSR protocols, and their results show that the proactive routing protocol, OLSR, performs well in most cases, except, according to [37], that it uses more bandwidth. However, there are some variations in the results shown between AODV and DSR in different literature. Although studies [36, 37] show that DSR is generally better than AODV, in [36], AODV is shown to perform better compared to DSR in more demanding scenarios[1]. Furthermore, AODV is shown to induce lower delays for most scenarios except the scenarios with varying speeds up to 20m/s in [36]. However in [37], the delays and jitter are higher in AODV compared to DSR when the number of traffic flows is less than around 50, but lower for other cases. The results in [37] indicates that AODV tends to perform better than DSR, except it has a higher routing load when varying the number of traffic flows. These conflicting results presented in different literature can be due to the following factors:

[1]This includes higher traffic loads (i.e. greater than around 75kbps), more number of traffic flows (i.e. more than about 17 traffic flows) and higher mobility (i.e. more than about 12m/s)

1. Different network simulation tools[2];
2. Different mobility models and parameters;
3. Different simulation parameters such as field size, network size, transmission range, average speed of the nodes and pause time;
4. Different traffic parameters such as traffic types, packet sizes, traffic loads, number of traffic flows, traffic duration; and
5. Different routing protocol parameters.

Given these differences in research studies performed by different people for different purposes, the conclusions drawn from the routing performance results of different studies can become conflicting, and hence, inconclusive. Furthermore, since there are no error bars in any of the comparison studies mentioned above, it is difficult to judge the confidence level of the results.

As suggested in the evaluation study of a number of routing protocols in [17], the proactive routing protocols tend to use more bandwidth due to the high overhead for route updates, but have lower delays as the routes are pre-established. Whereas for reactive route protocols, less bandwidth is required, but its trade-off is longer delays and jitter. Hence, there is no clear best type of routing protocol.

2.2.3 Distance Vector Routing Protocols

In typical distance vector routing protocols such as DSDV and AODV, the Bellman-Ford algorithm [38] is commonly used to compute the shortest paths. In a distance vector routing protocol, each node maintains a record of the distance (i.e. the number of hops) and the next node to each valid destination, and every node is required to inform its neighbours periodically of its own status and routing information for maintenance. The advantage of using distance vector is that it is simple to implement and compute, and hence requires less power and memory. Also, it only requires to send updates to its immediate neighbours. However, this means that each node only has information about its neighbours,

[2][36] used OPNET and [37] used NS-2

which may cause count-to-infinity problems to occur if there are link failures. Furthermore, distance vector may not scale well due to the long latency and the flooding of packets that are required to discover a new route after a link failure.

2.2.4 Link State Routing Protocols

Dijkstra's shortest path algorithm is commonly used for link state routing like OLSR. In link state routing protocols, every node maintains its own view of the whole network topology. As opposed to distance vector, link state routing protocols need to inform all the nodes in the network of any topology changes, thus, it floods its routing information (i.e. its link state) to the whole network. Though link state routing may be more expensive to implement compared to distance vector due to its computational complexity, it is more resilient to routing loops and it tends to be more scalable[3] compared to distance vector if proactive routing is used for both cases, as new routes can be found with relatively short convergence time, though flooding of the link state information is required. However, the scalability[4] of a MANET routing protocol not only depends on whether the protocol is distance vector or link state, but other factors such as reactive or proactive, and flat or hierarchical, as discussed in Section 2.2.6.

2.2.5 Flat

A flat routing topology is when all the nodes in the network are "peers" to one another, i.e. all the nodes can exchange routing information the same way. The advantage of using flat routing is that it is simple to implement. However, flat routing protocols are generally not scalable and routing overhead traffic can be quite high, and thus, it is suitable for a small network with low mobility. AODV,

[3]A MANET routing protocol is considered to be more scalable if the number of nodes in the network can increase while the routing performance is comparatively good.

[4]Scalability is the ability of a routing protocol to support more nodes with good routing performance.

DSR, DSDV are all classified as flat routing protocols.

2.2.6 Hierarchical

A hierarchical routing topology consists of two or more levels of layers of nodes in clusters. Such routing protocols, like in Zone Routing Protocol (ZRP) [39] and Fisheye State Routing (FSR) [40], allow some selected nodes to be cluster heads to form a hierarchy of clusters. All the nodes will maintain only local routing information within its own cluster. Only the cluster heads are able to exchange routing information between clusters. This can reduce the routing overhead traffic, and can support better scalability, but such routing protocols can be complex. Another advantage of using a hierarchical structure or clustering is that a choice of either proactive and reactive routing schemes can be made on different hierarchy levels, such hybrid protocols can take the advantages of both proactive and of reactive routing. However, the drawbacks are that there may be problems with the exchange of information, and that the problem of electing a cluster head can be difficult.

2.2.7 Routing Protocols with Prediction

As noted earlier, there is no single routing protocol that is suitable for all environments [17–24]. Over the last decade, researchers have started to explore the use of prediction to enhance MANET routing. Research on using mobility predictions in MANET routing has shown improvements in routing performance over the traditional MANET routing methods [2–12]. However, many of these studies show little improvement in performance compared to conventional MANET routing protocols, and only simple and unrealistic mobility models were selected for simulating and evaluating the routing protocols in the research studies. Moreover, the mobility of the nodes can affect the accuracy of the prediction [4], which affects the routing performance [41, 42]. Hence to date, there is not yet enough evidence to conclude that mobility predictions can effectively improve routing performance when applied on real MANETs.

2.3 Mobility Models

As shown in [41–44], the mobility of the nodes can affect the outcome of the network simulations. Thus, a variety of mobility models have been proposed in an attempt to imitate the "real" mobility of the mobile nodes for network simulations. These mobility models can be classified into two categories: the traces-based model; and the "synthetic"-based model [41]. In a traces-based model, the mobility information is obtained based on a real system. For "synthetic"-based model, the mobility of the nodes is computed using probability distributions and statistics.

The "synthetic"-based models can be further classified into two types: the entity mobility model (EMM); and the group mobility model [42–44]. In an EMM, the nodes' movement are independent of one another. Some examples of EMMs include Random Walk Model (RWM), Random Waypoint (RWP) Model and Gauss Markov Model (GMM). Whereas for group mobility model, the nodes move in groups, so the mobility of the nodes within the same group depends on the mobility of that group.

Three of the commonly used EMMs are selected for this thesis. RWM and RWP model are chosen for this research because they are widely used for modelling the mobility of mobile nodes. However, the mobility characteristics of these models are similar to each other, thus GMM is selected for its differences in mobility characteristics compared to RWM and RWP model. These three mobility models are described in this section.

2.3.1 Random Walk Model

The Random Walk Model (RWM) is one of the most basic form of random mobility models that can be used for simulating random motions [42–44]. In a basic RWM, a node begins at a randomly chosen location within a simulation field boundary, it then moves at a randomly selected speed towards a randomly selected direction for a fixed period of time, T, or a fixed distance, D. When the time is up or the node has travelled the required distance, the node will change direction and move in the new direction for duration T or distance D at another chosen speed. This process continues until the simulation ends.

Note that with the traditional RWM, if a node touches the boundary of the simulation field, the node "bounces" off the boundary in a direction depending on the incoming angle. However, in this study, the RWM is modified so that the node will never touch the boundary. This is achieved by discarding all the directions that lead to the node hitting the boundary, and use only the direction that will ensure the node does not touch the boundary for duration T or distance D, depending on the implementation that will be used. However, this type of mobility model is characterised as an "unrealistic" model due to the "sharp turns" and "sudden stops" in the mobility model [42].

If the duration T or the distance D of an RWM is short, the node changes direction more quickly, hence the node remains very close to the original location. On the other hand, if a long duration or distance is used, the network topology may become more dynamic.

In terms of prediction, it is expected that the RWM can be predicted more accurately using a deterministic prediction method due to its constant velocity motion, except when there is a change in direction. Therefore, the longer the duration or distance is, the more accurate the prediction will become. However, if the duration or distance is very short, the network topology becomes more static. In which case, using deterministic prediction techniques may induce great errors as the nodes continually changing directions while the network topology remains relatively stable.

2.3.2 Random Waypoint Model

Random Waypoint (RWP) Model was initially used by Johnson and Maltz [45], and is widely used in MANET research studies [3, 7, 10, 42–44, 46, 47]. In RWP models, a node moves in one direction at the same speed until it reaches the destination. When the node arrives at the destination, it pauses for a time randomly selected from a specified interval. Once the pause time is up, the node will move towards a newly chosen destination at a different speed within a given range of speeds. Hence the "sudden stops" issue is also in this mobility model. However, this model is a little more realistic compared to RWM, as pause time is introduced.

The motions of the RWP model is affected by both the speed and the pause time. When RWP model is used, the network topology can be more stable if either the pause time is long, or if the speed is slow.

Since the nodes either move at a constant velocity or not moving at all, it may be more appropriate to use deterministic prediction for this mobility model.

2.3.3 Gauss Markov Model

Gauss-Markov Model (GMM) was originated by Liang and Haas [48], and has been used to model nodes' mobility in other research studies [10, 41–44, 46, 49]. In GMMs, a node moves in a weighted random speed and direction towards a selected destination at every step it makes. Once a destination is reached, the node picks another destination to move towards with random speed and direction for each step. If the step is outside the boundary, which is quite possible when a GMM is used, the node will pick a new destination and begin to move towards that new destination.

This model was designed to introduce some randomness to the node's movement. This is achieved by using the following equations to determine the speed, s_n, and direction, d_n, of a node at the n^{th} instant:

$$\begin{aligned} s_n &= \alpha s_{n-1} + (1-\alpha)\bar{s} + \sqrt{(1-\alpha^2)} s_{x_{n-1}} \\ d_n &= \alpha d_{n-1} + (1-\alpha)\bar{d} + \sqrt{(1-\alpha^2)} d_{x_{n-1}} \end{aligned} \tag{2.1}$$

where $\alpha \in [0,1]$ is the tuning parameter for varying the randomness; $\bar{s}$ and $\bar{d}$ are the mean value of the speed and direction; and $s_{x_{n-1}}$ and $d_{x_{n-1}}$ are the random variables from a Gaussian distribution.

Therefore, the next position (x_{n+1}, y_{n+1}) of a node can be computed by the following equations:

$$\begin{aligned} x_{n+1} &= x_n + s_n \cos d_n \\ y_{n+1} &= y_n + s_n \sin d_n \end{aligned} \tag{2.2}$$

where (x_n, y_n) are the x and y coordinates of a mobile node's position at the n^{th} instant.

In GMMs, the "sudden stops" problem can be resolved by selecting the next velocity according to the current and the past velocities, hence it can be more realistic. The motion is more random in GMMs, hence more difficult to predict accurately using deterministic techniques. Thus, stochastic or heuristic techniques may be more appropriate. A discussion on the different types of prediction techniques is given in Section 2.4.4.

2.4 Prediction

The mobility of the nodes and the wireless medium have posed some challenges to routing performance of MANETs. As mentioned in Chapter 1 that prediction may increase the lifetime of the routes, which in turn may improve the routing performance. The existing prediction approaches proposed in the literature for MANETs will be discussed in this section.

Harri and others in [50] have identified a variety of prediction methods that have been proposed for wireless networks (including cellular networks, access point based WiFi networks and MANETs). However, many proposed prediction methods have not been tested on MANETs, and some may not even be suitable for MANETs. One reason may be because a MANET may consist of mobile nodes that have limited computational power and/or memory, and some of the prediction methods that were developed for cellular networks or wireless LANs may require a high computational complexity or a large database to store historic mobility records. Another reason is that some prediction methods rely on historic mobility patterns which require a fixed point of reference that may not be feasible for MANETs. Or it could also be because some prediction methods rely on using some centralised nodes to make predictions. Nevertheless, some research studies have shown that prediction can be used to improve MANET routing [2–15], however a lot of these assumed that the predictions are highly accurate, and that the transmission range is constant, such assumptions are unrealistic.

In order to make predictions for routing, one needs to determine what types of network state information can be used for making predictions, what needs to be predicted, and how the network state information can be obtained and the predicted information can be used in the routing protocols to assist routing. As some information can be more difficult to obtain, some information may be inaccurate, some may require more bandwidth to propagate the information through the network, and others may be difficult in terms of making accurate predictions. These issues are discussed in this section.

2.4.1 What Network State Information to Use?

Network state information consists of parameters that describe the state of a network at any one time. This includes the location and the velocity of the nodes, the Received Signal Strength (RSS), the Signal-to-Noise Ratio (SNR) and the link state. In order to make future predictions for routing, one or more of these parameters described in this section can be used.

2.4.1.1 Location and Velocity

The location and the velocity of the nodes can be used to calculate the distance between any two nodes. This information can be used to predict the future locations of the nodes, as well as the future distance between the nodes. Hence, one can use this information to either predict the future link state or the link lifetime.

Due to the increase in the popularity of Global Positioning Systems (GPS), nodes' location information are commonly assumed to be obtained from GPS [2–4, 6, 9, 51–63]. This is particularly true in the case of military platforms for which location is of paramount importance. However, this information may not always be available, and another issue is that link availability is not entirely dependent on the distance between two nodes, as a link can be affected by fading or interference.

2.4.1.2 Received Signal Strength

The Received Signal Strength (RSS) measures can provide indication to the connectivity strength between two nodes. The RSS can be obtained by measuring the power level of the signal received, and it can be useful in circumstances when the location information is not available. The studies presented in [64, 65] have suggested that various techniques can be used to estimate the current locations of the nodes using RSS. The estimated locations may then be used to predict the future link state or link lifetime. However, many of these RSS-based location estimation techniques are inaccurate [65–67].

In [68], Wu and others have used the transmitted signal strength, P_T, and the RSS, P_R, to estimate the distance, d_i, between two nodes using Equation 2.3. From a series of estimate distances, the relative velocity of the two nodes can be calculated, which can then be used to predict the lifetime of the link. However, this method requires the receiver to know the transmitter's signal strength, which may not be feasible.

$$d_i = \sqrt{\frac{P_T}{4\pi P_R}} \tag{2.3}$$

Paul *et al.* in [69] have proposed a scheme to periodically evaluate the "affinity", i.e. the strength of a link, between two nodes using a series of RSS. This means if the "affinity" value is high, the link is strong, and the lifetime can be long, and vice versa. The "affinity" value has been applied in routing [69, 70] in an attempt to select a stable route. However, the authors in [70] found that the "affinity" estimation is not very accurate.

2.4.1.3 Signal-to-Noise Ratio

The Signal-to-Noise Ratio (SNR) of a link can be measured by the receiver node. This information can be used to predict the future SNR values to estimate the reliability of a link. In [15], the SNR values are predicted using a time series of SNR values that are filtered by using a Kalman filter to eliminate the noise of the data. The results indicate that the accuracy of the predicted SNR values are affected by the mobility patterns of the nodes. The study suggests that for mobility models where mobility patterns are more predictable, the prediction accuracy drops to a threshold if the mobility is high. And if the mobility patterns are more random, the prediction accuracy reduces linearly as mobility increases.

2.4.1.4 Link State

The link state is the status of a link's availability. A link state can either be on for connected or off for disconnected. This information can easily be obtained. A link can be considered to be available if a transmission frame is received successfully. In studies [71–73], Jiang and others have proposed an algorithm to predict whether a link will be available continuously for a fixed period of time time T_p given that a link is currently available. However, this method of estimating link availability is not very accurate.

2.4.2 What to Predict?

Traditionally, a routing protocol selects a route based on the link's availability of all the links along the route. Therefore if one can predict if a link is available at a future time, or how long a link will remain available, the routing protocol can use this information to select a more long-lived route. This can achieved by making location or distance predictions, link availability predictions, or link duration or lifetime predictions.

2.4.2.1 Location or Distance Prediction

The location of a mobile node or the distance between two mobile nodes can be predicted using different methods (these prediction methods are discussed in more detail in Section 2.4.4). The predicted location or distance between two nodes are often used to estimate either the link availability or the link lifetime for making routing decisions. For example, the authors in [74] designed a Neural network to predict a series of future distances between two nodes to estimate the link lifetime.

2.4.2.2 Link Availability Prediction

The predicted link availability can be used for route maintenance. The future link availability can be predicted using one or more of the current and/or the past network state information explained in Section 2.4.1. Future link availability may be estimated using the link state information, like the method proposed in [71–73].

2.4.2.3 Link Duration or Lifetime Prediction

Link duration or lifetime prediction can be used to select a more long-lived route. The duration or the lifetime of a link is often predicted using the location and velocity information of the nodes [3, 75], the relative distance between two nodes [76], or the relative velocity of two nodes [77].

2.4.3 Prediction Approaches

Prediction approaches are ways in which predictions are made. Using the observed network state information, future network state can then be predicted by using proactive and/or reactive approaches described in this section.

2.4.3.1 Proactive Approaches

Proactive approaches can be used to describe prediction methods where a time series of nodes' locations or link state information is required to make predictions. Each node has to proactively transmit information to other nodes at regular time intervals. The advantage of this is that the nodes in the network have the knowledge of the whole network, and thus can predict the network state for the entire network and make routing decisions accordingly. Such methods can be suitable for proactive routing protocols. However, information needs to be broadcast at a regular interval, which may consume a lot of bandwidth unnecessarily if the relative mobility between the nodes is low, and the volume of data traffic is low. Also, backup strategies may be required for cases where location information is missing or delayed. One example of proactive prediction method used for MANET routing is proposed in [15], where a series of SNR values are collected to predict the future SNR values. In another example presented in [74], Kaaniche and Kamoun have proposed to use two series of locations of two nodes to predict the future distances between two nodes.

2.4.3.2 Reactive Approaches

Reactive approaches include methods that can make predictions without a time series of information. Information can be transmitted anytime by attaching it to data packets, routing packets or other control packets. This can save bandwidth, but the predictions may be less accurate if the volume of traffic is low, and the nodes may not know the state of the whole network, in which case, routing decisions can only be made based on this partial view of the network. The study in [4] is an example that utilises reactive prediction method in MANET routing. In that study, the authors used the transmission range, current locations, speeds and directions of the nodes to predict the Link Expiration Time (LET) of any two nodes. This method only requires the current knowledge to predict when the link-change will be. This type of method is useful for making route decisions during route discovery, so that longer lasting routes can be chosen. However, this requires the prediction to be accurate. If the prediction is inaccurate, it may cause the routing protocol to perform more poorly compared to when there is no prediction.

2.4.4 Prediction Techniques

Since predictions can assist routing protocols to select better routes provided that the predictions are accurate, a number of mobility prediction methods have been proposed for wireless networks [3, 12, 15, 49, 78–83]. In general, prediction techniques can be classified into the following categories:

1. **Deterministic** - assumes Constant-Velocity or Constant-Acceleration;
2. **Stochastic** - includes Markov Models, Kalman Filter and Particle Filter;
3. **History-based**; and
4. **Heuristic** - includes Artificial Neural Networks and Evolutionary Algorithm.

2.4.4.1 Deterministic

Deterministic is the simplest form of prediction technique. It makes predictions based on the observed location, velocity and/or acceleration of the target node, and assumes that the target node moves at a constant velocity or a constant acceleration. In [49], the authors have compared the prediction accuracy between the proposed prediction schemes that uses only the location information, with velocity information assuming constant velocity, and with acceleration information which assumes constant acceleration. Their results showed that the prediction scheme with velocity information available can make more accurate predictions.

A number of research studies have proposed using deterministic mobility prediction models in routing protocols to predict future positions. Their methods use the current location, speed and direction of a node to calculate its future location by assuming either constant velocity or constant acceleration. However, deterministic prediction methods are not very accurate for fast and dynamic movements. The studies in [7, 10] have shown that using deterministic prediction techniques can reduce the packet delivery ratio, end-to-end delay and routing overhead traffic load. Nevertheless, these studies did not test their protocol with highly dynamic motions using different mobility prediction techniques.

In [2–4], Su, Lee and Gerla proposed the use of GPS location and mobility information to predict the Link Expiration Time (LET) between the nodes along the paths for selecting the most stable routes for unicast and multicast routing protocols. A new route can be setup in advance just before the LET to handover before the link breaks. Their studies have shown that their proposed routing protocols with mobility prediction can reduce the routing overhead traffic and packet loss if the prediction is accurate. However, they did not show whether their protocols can reduce end-to-end delay, which is also an important routing performance metric. Their studies also show that routing performance degrades with short waypoint distance and with high mobility. This indicates that these predictive routing protocols require some degree of accuracy, meaning that it may not be suitable for some real life situations like driving on a winding road.

A number of other researchers have proposed different routing algorithms using the LET prediction method [6, 9, 11]. Yet these studies have not assessed their routing algorithms with different mobility models and other prediction methods to see how the accuracy of predictions may affect the routing performance, and did not compare the performance of different types of predictive protocols with different network sizes, density and rate of change.

2.4.4.1.1 Constant-Velocity

Constant-Velocity Based Prediction (CVBP) is a deterministic prediction approach where the velocity of a node is assumed to be constant. This means that the next position of the node, d_{t+1}, is defined by summing the current position of the node, d_t, and the multiplication of the current velocity, v_t, and the time difference, Δt, between the current and the next position as follow:

$$d_{t+1} = d_t + v_t \Delta t. \tag{2.4}$$

2.4.4.1.2 Constant-Acceleration

Constant-Acceleration based prediction technique assumes that the nodes move with a constant acceleration. Given the current acceleration of the node, a_t, the next position of the node, d_{t+1}, can be defined as:

$$d_{t+1} = d_t + a_t \Delta t^2. \tag{2.5}$$

2.4.4.2 Stochastic

Stochastic is a probabilistic prediction technique that includes techniques such as Autoregressive Model, Regression Model, Markov Model, Kalman Filter, and Particle Filter or Monte Carlo Method. Autoregressive Model has been used to predict the location of mobile nodes [84, 85]; Regression Model can be used to predict the duration of the link [86]; Markov Model can be used to predict link availability [87–89]; Kalman Filters are commonly used to predict the state (i.e. location, velocity, acceleration and/or direction) of the mobile nodes [56, 76]; and Particle Filters have been used to track the state of the mobile nodes [90, 91]. However, the researchers in these literatures have not tested their stochastic prediction methods on MANET routing protocols.

Stochastic prediction methods have been used to estimate a future node's position or link availability, and have been extensively used in areas such as target tracking and traffic prediction. However, only a few researchers have chosen to apply stochastic prediction methods to MANET routing mobility predictions. Although stochastic prediction techniques are not as commonly used in MANET routing as deterministic prediction techniques, a few research studies have explored the possibility of using stochastic prediction method in MANET routing. Farkas, Hossmann and others in [14, 15] have shown that the Kalman Filter can utilise SNR to predict the link quality for MANET routing, and has evaluated the accuracy of the prediction with different mobility models, but they did not show whether this method can improve routing performance over other routing protocols.

Similarly, the study described in [13] used signal strength as the distance between the nodes to predict the connectivity between nodes using a Markov model, but failed to show its advantages over other MANET routing techniques.

Creixell and Sezaki's work [12] has indicated that using an autoregressive process with a Least Squares Lattice Filter can predict the future location of the nodes for making routing decisions, and that their proposed protocol performs better than the Greedy Perimeter Stateless Routing protocol and the ellipsoid protocol in packet delivery ratio. They did not evaluate their protocol with other important routing performance metrics such as end-to-end delay and routing overhead traffic ratio. Although they claimed that their proposed protocol is independent of the mobility that is caused by varying the pause time, they did not test their protocol with other highly dynamic mobility models.

Despite that, stochastic prediction methods are well developed and extensively studied by researchers in other fields, very few have compared the performance of MANETs routing using stochastic prediction methods. Hence there is a need to explore how well stochastic prediction methods perform in MANET routing comparing to existing routing protocols.

Han *et al.* in [78] proposed a Link availability-Based Routing Protocol (LBRP). This protocol predicts the duration of the previous link and selects the path with the best path availability based on the predicted link quality using a stochastic prediction method. However, the authors have indicated that this technique can only predict the link availability accurately "over a short period of time". Hence this may not be suitable for networks with high mobility nodes. Furthermore, the paper has made a routing performance comparison with other routing protocols such as AODV, DSR and ZRP in terms of percentage of packet delivery, end-to-end delay and the number of messages per delivery over traffic flows and network size. This work did not show how the accuracy of predictions may affect the routing performance of the LBRP.

2.4.4.3 History-based

History-based techniques have been widely used to predict users' mobility in cellular networks [79–82]. Such techniques allow the network to accurately predict when the users will switch to another base station. Often, History-based techniques are required to keep a record of some common user mobility patterns, so that regularly used paths can be predicted accurately. However, these techniques mostly require a fixed base station or an access point present in the network to function as a point of reference. Unfortunately, there is no regular mobility pattern for MANETs, as the motion of the nodes in a MANET is dynamic, and MANETs are infrastructureless and hence may not have a fixed non-mobile node. Therefore, such techniques are most likely unsuitable for MANETs.

2.4.4.4 Heuristic

Heuristic techniques have also been proposed for mobility prediction in wireless networks. Neural networks have been proposed for node mobility prediction in [92, 93], and Mala *et al.* [83] have proposed to use Genetic algorithm for mobility prediction. However, these techniques were designed for cellular networks.

It is less common to use heuristic techniques for mobility prediction. This may be due to their higher demand of computational power, and their requirement of using historic records for training. Nevertheless, Kaaniche and Kamoun [74] have proposed to use a Neural network to predict the next N future locations of each MANET node, which are then used to estimate the LET of the link between any two nodes by determining the time the distance between two node is larger than the range. However, the number of future locations predicted is fixed, so this technique may not be suitable if the LET is longer than N. Furthermore, it is uncertain how useful it is when it is applied in routing, as the authors did not compare the this technique with other techniques, and have not tested it on any

MANET routing protocols.

2.5 Conclusion

This chapter has provided an overview of the advantages and disadvantages of different types of MANET Routing Protocols, some common EMMs used in other MANET research, and different types of prediction methods used in MANET routing research.

Chapter 3

Prediction Methods and Evaluation

3.1 Introduction

As mentioned in Chapter 2, research studies have shown that routing with mobility predictions can provide some advantages over conventional routing methods with the assumption that the predictions are accurate. However, routing with predictions may perform poorly if the prediction accuracy is low. Therefore, it is important to explore how accurate the prediction is required to be in order to show improvements in routing performance.

In order to study the prediction accuracy and the evaluation techniques, it is important to evaluate the accuracy of mobility predictions using one or more evaluation metrics that are relevant to routing, such as the link states and the change of link states[1]. Hence this chapter compares the differences of three evaluation metrics for MANET mobility prediction, and applies these methods on different mobility models and different prediction methods to compare the different prediction methods. This is achieved with the assumption that the connectivity of any wireless link between two nodes can be determined by the

[1]The change of link states are referred to as the "link changes".

distance between the nodes.

Firstly, Section 3.2 provides an overview of existing work on mobility prediction, and introduces the three evaluation metrics that can be used for evaluating the accuracy of MANET mobility prediction. Section 3.3 compares the three evaluation metrics by evaluating the accuracy of predicting three different mobility models. Finally, an evaluation of the accuracy of different prediction methods using two of the three evaluation metrics is provided in Section 3.4.

3.2 Evaluation Metrics

In the literature, the accuracy of mobility prediction has been commonly evaluated by calculating the average Euclidean distance (or the Root Mean Square Error (RMSE)) between the real location and the predicted location [49, 51, 56, 57, 85, 91, 94, 95], between the real and the predicted Received Signal Strength (RSS) [64], or between the real and the predicted Signal-to-Noise Ratio (SNR) [15]. For example, Xu *et al.* in [49] have compared the Velocity-based and Acceleration-based predictions with the Random Waypoint and the Gauss-Markov mobility models using RMSE between the real and the predicted location of the nodes. Likewise, Zaidi and Mark [56, 57, 85, 95] have compared the accuracy of their Autoregressive prediction models using RMSE. Such metrics have been very useful for evaluating the accuracy for target tracking, and tracking for cellular networks. However, it fails to provide a meaningful indication of how accurate the predicted topology of the network or the predicted link changes are.

Alternatively, Sharma *et al.* in [5] have evaluated the prediction accuracy by counting the number of predicted grid numbers that are correctly predicted. This evaluation method can be useful for networks that use a grid system, but it may not be appropriate for all MANET routing methods. This is because having accurately predicted grid numbers does not mean that the link state of the nodes can be predicted.

MANET routing performance is affected by the network topology and the link changes. Hence predicting the Euclidean distance error and the grid counting methods may not be as practical for evaluating the prediction accuracy for routing. Routing performance is predominately affected by the link state and the link change. It is anticipated that link state and link change metrics may be more useful for evaluating the accuracy of mobility prediction for routing. However, Euclidean distance error, which can be used to represent the signal strength, may provide a more realistic view of the link quality as opposed the grid counting method. Thus for comparison, the following three evaluation metrics have been selected to evaluate the accuracy of mobility prediction:

1. Average Euclidean Distance Error;
2. Probability of Incorrectly Predicted Link State (IPLS); and
3. Probability of Incorrectly Predicted Link Change (IPLC).

3.2.1 Average Euclidean Distance Error

The average Euclidean Distance Error is the mean distances between the real location (x_R, y_R) and the predicted location (x_P, y_P) of each of the nodes in the network at a given time. The Euclidean Distance Error, d_{err}, is calculated by:

$$d_{err} = |L_R - L_P| = \sqrt{(x_R - x_P)^2 + (y_R - y_P)^2} \tag{3.1}$$

where L_R and L_P are the real location and the predicted location of a node respectively. Thus, the average Euclidean Distance Error over a time series of T steps is represented by:

$$E\left[d_{err}\right] = \frac{\sum\limits_{t=1}^{T}\left(\sum\limits_{n=1}^{N} d_{err}\left(n,t\right)\right)/N}{T} \tag{3.2}$$

where N is the total number of nodes in the network; and $d_{err}(n,t)$ is d_{err} for the n^{th} node at time t.

Evaluating the distance error can provide some insights about the prediction error in the link quality, which can be useful. However, as mentioned previously, calculating the distance error may not be appropriate for evaluating the accuracy of the predictions used for MANET routing. The reason is that routing performance is affected by the link quality which is not linearly dependent on the distance between the nodes. If two nodes are within the transmission range of each other, they are connected regardless of how close they are located. Likewise, if two nodes are out of range, they are disconnected regardless of their distance apart. In a perfect environment, link quality is a step function. Thus it may be more appropriate to measure the accuracy of prediction in terms of link state, as discussed in more detail in Section 3.2.2.

3.2.2 Probability of Incorrectly Predicted Link State (IPLS)

IPLS describes a link state that is predicted incorrectly. The term "link state" is used to describe the state of a link such that if the link state between two nodes is on (= 1), the two nodes are in range (or connected)[2], and if the link state between two nodes is off (= 0), the two nodes are out of range (or disconnected).

The probability of IPLS at any one time, t, can be represented by the graph distance[3] [96], $d\left(G_R(t), G_P(t)\right)$, between the real topology or graph $G_R(t)$, a matrix that represents the real network topology at time t, and the predicted topology or graph $G_P(t)$, a matrix that represents the predicted network topology (that is predicted at time $t-\Delta t$) for time t. The graph distance can be calculated by the following equation:

$$d\left(G_R(t), G_P(t)\right) = 1 - \frac{|mcs\left(G_R(t) \cap G_P(t)\right)|}{|G_R(t) \cup G_P(t)|} \tag{3.3}$$

where $mcs\left(G_R(t) \cap G_P(t)\right)$ is the maximum common subgraph of $G_R(t)$ and $G_P(t)$. Therefore, the average probability of IPLS can be evaluated by the

[2]Connected means data can be exchanged reliably.

[3]Graph Distance is a measure to evaluate the difference between two or more graphs.

following equation:

$$P\{IPLS\} = E\left[d\left(G_R(k), G_P(k)\right)\right] = \frac{\sum_{k=1}^{K} d\left(G_R(k), G_P(k)\right)}{K} \tag{3.4}$$

where k represents the k^{th} time step where predictions were made; and K is the total number of predictions made.

3.2.3 Probability of Incorrectly Predicted Link Change (IPLC)

Link change is one of the most important parameters that affect routing performance. If one can predict the link change accurately, routing issues that are caused by nodes' mobility and connectivity can be solved. The authors in [97] have evaluated the accuracy of cluster change prediction with different prediction methods for cellular networks and cluster based networks. However, this study only focuses on the cluster change in cellular and cluster based networks, the accuracy of predicted link changes for MANETs with different mobility models remains unexplored. Therefore in this study, the prediction accuracy in terms of link change for MANETs with different mobility models has been evaluated in comparison with the accuracy of link state prediction. This section explains how the accuracy of predicted link changes in a MANET will be evaluated, which will be referred to as the Incorrectly Predicted Link Change (IPLC).

Link change describes the change of link state between the previous time step, $t-\Delta t$, and the current time step, t. This means there is a link change if the link state at time $t-\Delta t$ is different to the link state at time t, and there is no link change if the link states are the same. Therefore, the relationship between link state Ls and link change Lc can be represented by the following equation:

$$Lc(t) = Ls(t-\Delta t) \oplus Ls(t) \tag{3.5}$$

Similar to IPLS, the probability of IPLC at time t can be evaluated measuring the similarity between the matrix representing the real link changes of a network from time $t-\Delta t$ to time t and the matrix representing the predicted link change of a network predicted at time $t-\Delta t$.

In relation to the network graph of real link state, the graph of real link change representing all the link changes that occur from time $t-\Delta t$ to time t is denoted by the following equation:

$$\begin{aligned} C_R(t) &= G_R(t-\Delta t) \oplus G_R(t) \\ &= \left|\left|G_R(t-\Delta t) \cap G_R(t)'\right| \cup \left|G_R(t-\Delta t)' \cap G_R(t)\right|\right| \end{aligned} \tag{3.6}$$

The graph of predicted link change representing all the links that have been predicted at time $t-\Delta t$ to be changed at time t is denoted by the following equation:

$$\begin{aligned} C_P(t) &= G_R(t-\Delta t) \oplus G_P(t) \\ &= \left|\left|G_R(t-\Delta t) \cap G_P(t)'\right| \cup \left|G_R(t-\Delta t)' \cap G_P(t)\right|\right| \end{aligned} \tag{3.7}$$

Thus, the graph distance between $C_R(t)$ and $C_P(t)$ at time t is given by the following equation:

$$d\left(C_R(t), C_P(t)\right) = 1 - \frac{\left|mcs\left(C_R(t) \cap C_P(t)\right)\right|}{\left|C_R(t) \cup C_P(t)\right|} \tag{3.8}$$

And the average probability of IPLC is:

$$P\{IPLC\} = E\left[d\left(C_R(k), C_P(k)\right)\right] = \frac{\sum_{k=1}^{K} d\left(C_R(k), C_P(k)\right)}{K} \tag{3.9}$$

3.3 A Comparison of the Evaluation Metrics

This section compares the three evaluation metrics described in Section 3.2. In order to achieve this, a number of simulations (described in Section 3.3.1) have been executed to explore the different prediction accuracy evaluation metrics, and what the evaluation metrics say about the prediction accuracy of different mobility models (discussed in Section 3.3.1). Since the simulations are intended to study the differences between the three evaluation metrics described in Section 3.2, there is no need to implement any protocols within the nodes or any traffic between the nodes. Also, the simulations implemented only consist of nodes moving in a fixed 2-dimensional field, with the following assumptions being made:

1. The state (location) information about all the mobile nodes in the simulation at any given time is known immediately - "The ideal observer" - i.e. no time is required to transmit that information;
2. The connectivity of the nodes will be determined by the distance between the nodes;
3. The links are bidirectional; and
4. All the location and velocity information about the nodes are accurate.

The assumptions above were made to artificially create an ideal simulation environment. This ensures that the errors obtained in the results will purely be induced by the errors in the prediction, rather than by other factors such as fading, uni-directional links, missing or inaccurate location information.

3.3.1 Simulation Methodology

A program for generating the mobility patterns and the predicted location of the nodes on a field has been written in a JAVA package called MASON [98], which is a discrete-event multiagent simulation package. MATLAB [99] is then used to analyse the generated data from MASON. MATLAB is a scientific software tool that is useful for algorithm computations, signal processing and data analysis.

The simulations used for this study are based on a fixed rectangular area of size $2400m \times 1800m$ with 25 mobile nodes, and the range of these mobile nodes are assumed to be $300m$. This means a link is considered to be available if the distance between any two mobile nodes is less then or equal to $300m$.

Three mobility models were chosen to model the nodes' movements - RWM, RWP model and GMM, as discussed in Section 2.3. These mobility models were chosen because they have different mobility properties. The prediction model that was chosen to predict the nodes' motion is the Constant-Velocity Based Prediction (CVBP) model, as described in Section 2.4.4.1. This was chosen because it is the most basic form of prediction method and it is easy to understand.

The minimum, maximum and average speeds chosen for the RWM and RWP models are 1.5, 4.5 and $3m/s$ respectively. The minimum, maximum and average pause times for the RWP model are 0, 60 and $30s$. And the parameters used for GMM are: pause time = $0s$; mean speed ($\bar{s}$) = $3m/s$; mean direction ($\bar{d}$) = 0 degree; tuning parameter (α) = 0.5; and the variance for both speed and direction (σ^2) = 1.

For each mobility model, separate simulations have been executed with different durations of prediction step intervals (or sampling intervals) - 1, 2, 5, 10, 20, 30, 50, 60, 80, 100, 150, 200 and $250s$. A prediction step interval is defined as the interval for which a prediction is made. For example, if the prediction step interval is 1 second, the prediction that is made at time t seconds, is predicting the location at time $t+1$ seconds. Similarly, if the duration of a prediction step interval is Δt seconds, the prediction made at time, t seconds, is predicting the location at time, $t+\Delta t$ seconds. Note that varying the step interval is equivalent to varying the node speed, hence only slow (walking/running) speeds are used, and that it is unnecessary to vary the node speeds.

Ten thousand steps will be executed for each simulation, i.e the duration of each simulation is $(10,000\times$ sampling interval) seconds. This means the duration of each simulations is different depending on the sampling interval used, but this ensures that 10,000 samples will be collected for evaluations.

3.3.2 Results and Analysis

Results obtained from the simulations described in Section 3.3.1 indicate that all three evaluation metrics differ in various ways. These results are discussed and analysed in this section in more detail.

The graph in Figure 3.1 shows the average Euclidean Distance Error between the real and predicted location of each node for GMM, RWM and RWP mobility models. The x-axis in Figure 3.1 represents the prediction step interval in seconds, and the y-axis indicates the average Euclidean Distance Error. The graph shows that the longer the step interval, the prediction error increases exponentially, this means that the prediction is less accurate if the step interval is large. The high Euclidean distance error for GMM indicates that it is much more difficult to predict compared to RWM and RWP models. This is because there is some randomness in the motion of GMMs, but CVBP assumes constant velocity. Whereas for both RWM and RWP models, the velocity and direction remains unchanged for a period of time, hence these RWM and RWP motions can be predicted more accurately with CVBP. Also keep in mind that with all three mobility models, the Euclidean Distance Error increases exponentially as the step interval increases.

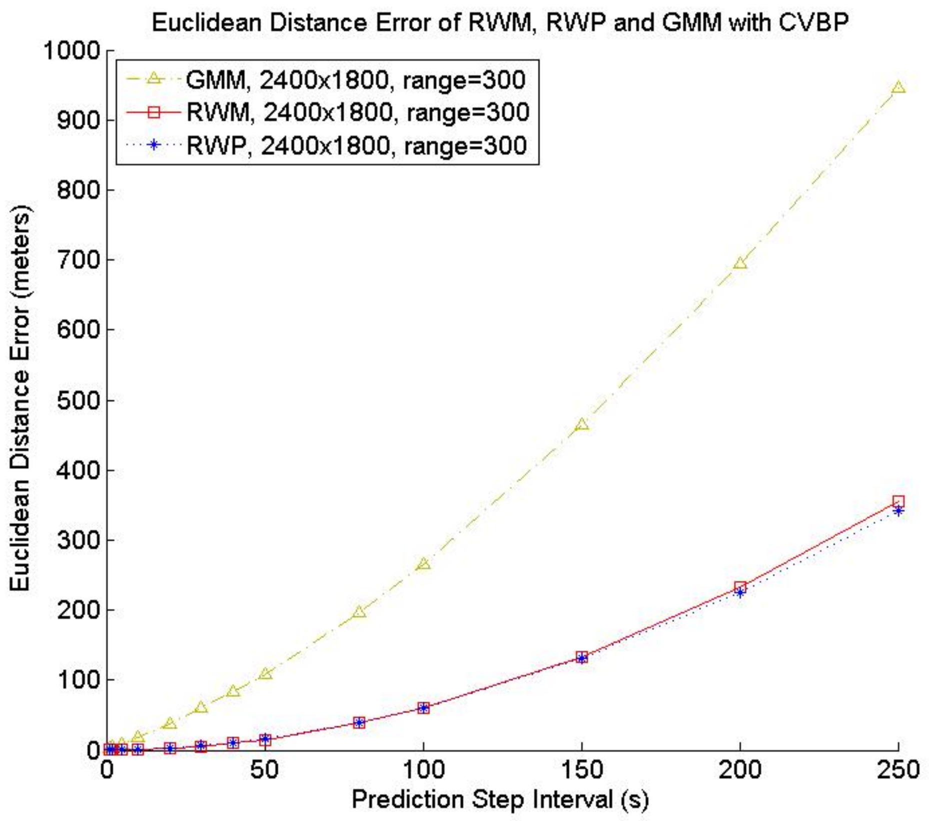


Figure 3.1: *Average Euclidean Distance Error for GMM, RWM and RWP mobility models using CVBP method over a range of step intervals.*

Figure 3.2 represents the probability of IPLS between the real and the predicted network topologies for GMM, RWM and RWP mobility models against different duration of step intervals. This plot scales from 0 to 1, with 0 meaning perfect accuracy, and 1 means no link is predicted correctly. Each data point on the plot is obtained by averaging over five simulations with different seeds, and each simulation was executed over a time series of ten thousand steps, and was calculated by using Equation 3.4 on page 35. The graph in Figure 3.2 indicates that for small prediction step intervals, the link state predictions were quite accurate for all three mobility models, but for large prediction step intervals, the probability of IPLS approaches one, i.e. almost all the link states are predicted incorrectly when the step interval is large. The graph also shows that GMM becomes less accurate in link state prediction at a faster rate in comparison to RWM and RWP models, which agrees with the average Euclidean distance error graphs. This is expected, as GMM's motion is less likely to be predicted by using CVBP. Note that because IPLS has a bounded scale between 0 to 1, the prediction error saturates at 1 rather than increases exponentially like the Euclidean Distance Error. So using this metric, the difference between the accuracy of GMM and the other two models peaks at a step interval of around 100s, and gradually reduces for larger step intervals.

Due to the uncertainties in the confidence level of the results represented in Figure 3.2, to give an indication of the difference of the accuracy of using CVBP to predict RWM and RWP models, a graph with error bars is given in Figure 3.3. This graph shows the probability of IPLS for RWM and RWP models with a $2400m \times 1800m$ field and $300m$ transmission range with error bars indicating the confidence intervals of the IPLS measured at different prediction step intervals. The error bars on the graph indicate that the confidence intervals are smaller for small prediction step intervals, indicating that the results are more reliable when the step interval is short. However, the confidence interval grew as the prediction step interval increases. Since the accuracy of the prediction in terms of graph distance varies with large prediction step intervals, it is indicating less reliable results. This graph also indicates that the IPLS of both RWM and RWP models are similar to one another.

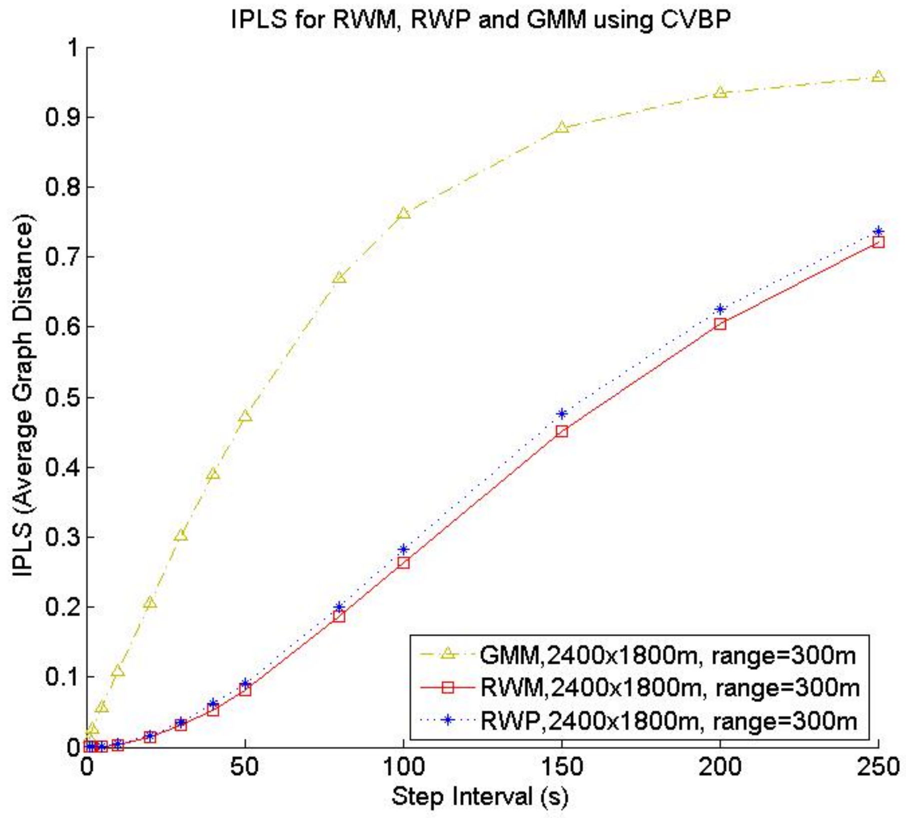


Figure 3.2: *Probability of IPLS for GMM, RWM and RWP mobility models using CVBP method over a range of step intervals.*

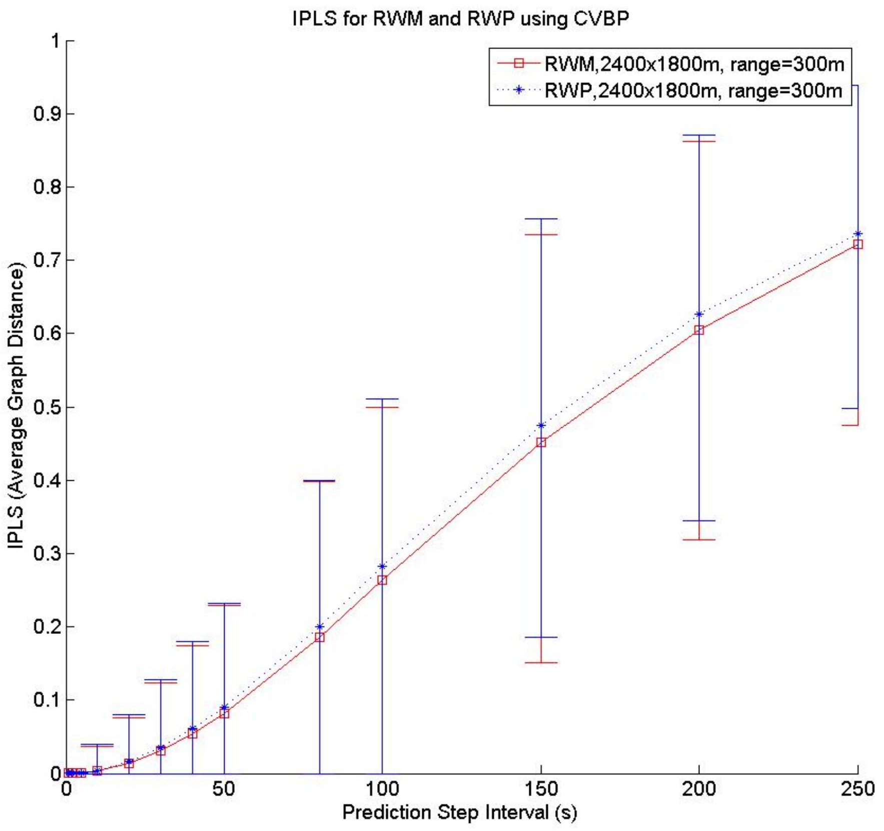


Figure 3.3: *Probability of IPLS for RWM and RWP mobility models with confidence intervals.*

The graph of probability of IPLC for RWM, RWP and GMM mobility models against the duration of prediction step intervals is shown in Figure 3.4. It shows that both RWM and RWP models are very accurate in terms of link change prediction when the step intervals are small, but the probability of IPLC increases to around 0.5 for large step intervals, meaning that the accuracy drops rapidly. The reason that the probability of IPLC only reaches around 0.5 is because of the artifact of using a bounded simulation field and making unbounded predictions. When the step interval is small, the predicted locations are unlikely to be out of boundary, but if the step interval is large, the likelihood that the predicted locations are outside the boundary is large. Therefore, when the step interval is large, the real locations of the nodes at time t remains within the simulation area, but the locations of the nodes that were predicted for time t can be sparsely distributed outside the boundary, thus the probability that any two nodes are predicted to be connected is lower than the probability that the nodes are connected in the real simulation. This means for a very large step interval, all the existing links will be predicted to be disconnected at the next step, and no new links will be predicted (as the predicted nodes' locations are too far away from each other). But in the real simulation, new links will form while most of the old links will disconnect. Hence, for very large step intervals, only around 50% of the link changes can be correctly predicted.

Figure 3.4 also shows that the link changes of GMM is difficult to be predicted using a Constant-Velocity Based prediction model even for small step intervals. This is because a link change only occurs when the nodes are near the edge of one another, so a small error may affect the predicted link change dramatically, and the speed and direction of the nodes in GMM can vary at each time step. Thus it is very difficult to predict the link change of the GMM using a CVBP model. Due to the stochastic motion characteristic of GMMs, it is believed that stochastic prediction methods can make better predictions for GMMs, which is explored later in Section 3.4.

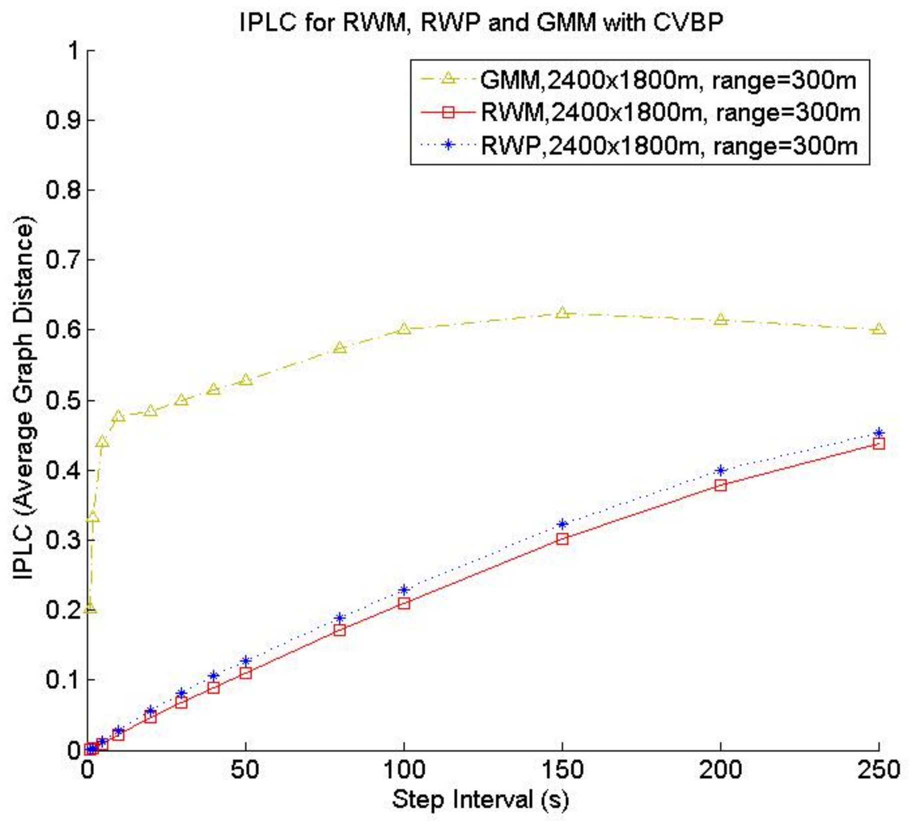


Figure 3.4: *Probability of IPLC for GMM, RWM and RWP mobility models using CVBP method over a range of step intervals.*

In this chapter, the results for large prediction step intervals are affected by the artifact of having a fixed number of nodes moving within a bounded simulation field while the prediction model is able to predict the future location of the node to be outside the simulation field. Although there have been studies addressing the effects of using different mobility models within a bounded simulation area on the performance of MANET routing protocols [42, 44], to the author's knowledge, there has been no literature discussing the implications of having a boundary affecting the prediction results. There is no known method that can completely overcome the boundary problem, as different models will cause different problems. For example, if the simulation field is extended or made unbounded, the nodes can be sparsely distributed that the connectivity between any nodes will be reduced. If the field is made to wrap around, the formation of the network is unrealistic. And if the predicted location is ensured to be within the boundary, the accuracy of the prediction would be affected. However, this is out of the scope of this research study.

Overall, the results show that RWM and RWP models can be predicted more accurately in comparison to GMM for all cases. This is because for both RWM and RWP models, the nodes travel at a constant velocity for some distance before changing speed and direction, whereas for GMM, the speed and direction change every second. When the step intervals are small, all three metrics show that RWM and RWP models can be predicted accurately, but GMM is only accurate in terms of Average Euclidean distance error and IPLS, and much less accurate when it is evaluated against IPLC even at very small time intervals. This is due to the stochastic properties of GMM, and because the chances of getting a link change is small, it is more difficult to predict link changes accurately.

It can be seen from the results that each of the three types of evaluation methods can make different conclusions in terms of the accuracy of the predictions. The Euclidean Distance Error measures the errors in terms of distance. Thus, this evaluation metric shows higher accuracy for mobility simulations with small step intervals or with slow mobility, but fails to show when the accuracy becomes too low for routing, and this evaluation method can only be used for prediction methods that make predictions in terms of location, distance, time or signal strength.

IPLS indicates the likelihood of predicting a link state accurately. Therefore, with this method high accuracy can be illustrated for small step intervals and/or slow mobility simulations, and low accuracy for simulations with large step intervals, high mobility and/or frequent change of directions. Whereas IPLC shows the probability of whether a link change can be predicted. Link changes are more difficult to predict compared to link states, because link changes occur rarely and they rely on accurate link state predictions. Thus, high link change accuracy can be achieved for mobility with less frequent or very small change of directions and with slow mobility. Since IPLS and IPLC appears to be better evaluation methods for MANETs, these two evaluation metrics are selected for evaluating the prediction accuracies for different prediction methods.

3.4 Evaluation of the Accuracy of Prediction Methods

In the previous sections, three different mobility prediction evaluation metrics have been discussed and compared by evaluating the accuracy of using CVBP method on GMM, RWM and RWP mobility models. However, CVBP is not the only prediction method available, and there are many prediction methods that have been shown to be more accurate for predicting more random motion. In this section, two other prediction methods are selected and compared against CVBP to show how different prediction methods are suitable for different mobility models. The accuracy of these prediction methods are analysed for different mobility models and evaluated using IPLS and IPLC evaluation metrics.

3.4.1 Prediction Methods

In order to compare the accuracy of different prediction methods, prediction methods include CVBP, Markov Chain link state prediction and Kalman Filter are used for this study. Details on the Markov Chain and Kalman Filter prediction methods are described in Sections 3.4.1.1 and 3.4.1.2 below.

The mobility models selected for this study are RWM and GMM. These two mobility models are chosen (with RWP model excluded) is because the results from Section 3.3.2 show very little differences in the prediction accuracies between the RWM and RWP models for all three evaluation metrics, while GMM's performance is very different. Furthermore, the motion of the RWM and RWP models are very similar. Hence, either RWM or RWP model can be selected for evaluating the different prediction methods. Since RWM is more similar to GMM, and it is likely to be more dynamic compared to RWP model as it has no pause time between changing directions, RWM and GMM are chosen for the evaluation of the prediction accuracies for different prediction methods.

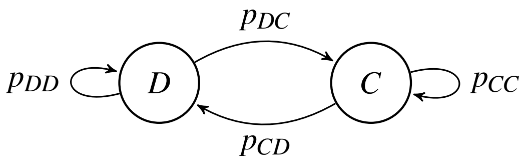
3.4.1.1 Markov Chain Link State Prediction Method

Studies in [87, 88, 100, 101] have shown that Markov Chains can be used for mobility predictions. In this study, the Markov Chain link state prediction method is implemented by using a discrete time Markov model proposed by Hwang and Kim in [88] that predicts the next link state between any two nodes in the network. This Markov model consists of two states (connected, C, and disconnected, D) as illustrated in the state transition diagram in Figure 3.5, where:

1. $p_{DC} = P\{X_k = C|X_{k-1} = D\}$ is the probability of any two nodes disconnected at time $k-1$ becomes connected at time k;
2. $p_{CD} = P\{X_k = D|X_{k-1} = C\}$ is the probability of a connected link at time $k-1$ being disconnected at time k;
3. $p_{CC} = P\{X_k = C|X_{k-1} = C\} = 1 - p_{CD}$ is the probability of two nodes

staying connected from time $k-1$ to time k; and

4. $p_{DD} = P\{X_k = D | X_{k-1} = D\} = 1 - p_{DC}$ is the probability of two nodes remaining disconnected from time $k-1$ to time k;.



$$\begin{tikzcd} D \arrow[loop left, "p_{DD}"] \arrow[r, bend left, "p_{DC}"] & C \arrow[l, bend left, "p_{CD}"] \arrow[loop right, "p_{CC}"] \end{tikzcd}$$

Figure 3.5: *Markov Link State Prediction Transition Probability Model Diagram.*

Hence, the transition probability matrix, P, can be written as:

$$\mathbf{P} = \begin{bmatrix} p_{DD} & p_{CD} \\ p_{DC} & p_{CC} \end{bmatrix} = \begin{bmatrix} 1-p_{DC} & p_{CD} \\ p_{DC} & 1-p_{CD} \end{bmatrix} \tag{3.10}$$

As time approaches infinity, the stationary matrix is:

$$\mathbf{Q} = \lim_{k\to\infty} \mathbf{P^k} = \begin{bmatrix} \frac{p_{CD}}{p_{DC}+p_{CD}} & \frac{p_{CD}}{p_{DC}+p_{CD}} \\ \frac{p_{DC}}{p_{DC}+p_{CD}} & \frac{p_{DC}}{p_{DC}+p_{CD}} \end{bmatrix} \tag{3.11}$$

Therefore, the probability a link is connected at time k, $p_C(k)$, and the probability that a link is disconnected at time k, $p_D(k)$, are:

$$\lim_{k\to\infty} p_C(k) = \frac{p_{DC}}{p_{DC}+p_{CD}} \tag{3.12}$$

$$\lim_{k\to\infty} p_D(k) = \frac{p_{CD}}{p_{DC}+p_{CD}} \tag{3.13}$$

A Markov Chain model is required to be trained before it can be used for making predictions. The training stage is required to compute the state transition probability matrix, which can then be used for prediction. For this Markov Chain model, the state transition probability matrix can be obtained by counting the number of times the link state remains at either state C and state D over n

time steps (during the training phase). Through simulation, the probability that a link is connected and disconnected can be computed using N_C/n and N_D/n respectively, where N_C is the number of times the link is connected, N_D is the number of times the link is disconnected, and n is the number of time steps. By substituting these values into Equations 3.12 and 3.13, the state transition probabilities can be calculated, and hence the link state can be predicted using the state transition matrix. After the training, the transition matrix is then used to make predictions for the remaining time, and during this time the state transition properties will continue to be updated at each time step during the prediction stage in an attempt to improve the accuracy of the prediction while predicting.

3.4.1.2 Kalman Filter Prediction Method

Kalman Filter [102, 103] is another stochastic prediction method that has been widely used for tracking. A linear Kalman Filter consists of a set of equations that are used to estimate the state vector of a discrete-time process, $x \in \Re^n$, where n is the number of real numbers in the vector x. The predicted state vector is represented as:

$$x_k = Fx_{k-1} + Bu_{k-1} + w_{k-1} \tag{3.14}$$

where F is the state transition matrix; B is the control-input matrix; x_{k-1} is the previous state vector; u_k is the control vector; and $w_k \sim N(0, Q_k)$ is the process noise with zero mean and covariance Q_k.

The measurement, $z_k \in \Re^m$ is:

$$z_k = Hx_k + v_k \tag{3.15}$$

where H is the observation matrix; and $v_k \sim N(0, R_k)$ is the observation noise with zero mean and covariance R.

There are two main processes involved in a Kalman Filter estimation process. At each time iteration, the Kalman Filter predicts the process state, and then it gets a feedback from the measurement.

1. **Predict:**
   This process consists of two equations. One predicts the current process state using the previous process state, and the other one predicts the covariance of the process noise, w.

   The predicted (*a priori*) state estimate vector, $\hat{x}_{k|k-1}$, at the k^{th} time step is given by:
   $$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1} + Bu_{k-1} \tag{3.16}$$
   The predicted (*a priori*) error covariance, $P_{k|k-1}$, is then given by:
   $$P_{k|k-1} = FP_{k-1|k-1}F^T + Q \tag{3.17}$$
   where $P_{k-1|k-1}$ is the previous error covariance matrix.

2. **Update:**
   This process is to update the state estimate using the current observed measurements and the predicted state estimate.

   The Kalman gain, K_k, is computed using:
   $$K_k = P_{k|k-1}H^T(HP_{k|k-1}H^T + R)^{-1} \tag{3.18}$$
   Update the state estimate, $\hat{x}_k$, using the measurements:
   $$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k(z_k - H\hat{x}_{k|k-1}) \tag{3.19}$$
   The error covariance matrix, P_k, is updated as follow:
   $$P_{k|k} = (I - K_kH)P_{k|k-1} \tag{3.20}$$

Note that the variables F, H, Q, R and P are unknown. Therefore in order to make predictions using this Kalman Filter, these variables need to be initialised.

In this study, Kalman Filter is used to predict the location and the velocity of the nodes. With the assumption that the acceleration is constant, the state vector of each node is represented by a linear state vector, $[x, y, \dot{x}, \dot{y}]$. The variables x and y represent the location of the node; and $\dot{x}$ and $\dot{y}$ represent the velocity of the node. Assuming that the sampling interval is S seconds, the variables F, H, Q, R and P are initialised as follow:

The state transition model F is:

$$F = \begin{bmatrix} 1 & 0 & S & 0 \\ 0 & 1 & 0 & S \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

The observation model H is:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{3.22}$$

The process noise covariance Q is:

$$Q = \begin{bmatrix} S^3/3 & 0 & S^2/2 & 0 \\ 0 & S^3/3 & 0 & S^2/2 \\ S^2/2 & 0 & S & 0 \\ 0 & S^2/2 & 0 & S \end{bmatrix} q \tag{3.23}$$

where q is the process noise intensity. The observation noise covariance R is:

$$R = \begin{bmatrix} R_x & 0 \\ 0 & R_y \end{bmatrix} \tag{3.24}$$

where R_x is the covariance of x; and R_y is the covariance of y. And the error covariance matrix P is:

$$P = \begin{bmatrix} R_x & 0 & R_x/S & 0 \\ 0 & R_y & 0 & R_y/S \\ R_x/S & 0 & 2R_x/S^2 & 0 \\ 0 & R_y/S & 0 & 2R_y/S^2 \end{bmatrix} \tag{3.25}$$

Note that the process noise intensity, q, and the covariance of x and y can be adjusted to tune the Kalman Filter to adapt to different mobility models. The values chosen for the variables q, x and y are discussed in Section 3.4.2.

3.4.2 Simulation Methodology

The simulations executed in this section are similar to that explained in Section 3.3.1. The difference is that in this section, the prediction accuracy of three prediction methods are evaluated and compared with two mobility models. In achieving this, GMM and RWM are selected to simulate the nodes' movements. As mentioned previously, the prediction methods chosen for this exercise are CVBP, Markov Chain and Kalman Filter.

All three prediction methods, CVBP, Markov Chain and Kalman Filter, are implemented in MATLAB. The mobility models and parameters that are used for the simulations in this section (Section 3.4) are the same as the ones in Section 3.3 on comparing the evaluation metrics.

The sampling intervals chosen to be evaluated are 1, 2, 5, 10, 20, 30, 40, 50, 80, 100, 150, 200 and 250s. For each sampling interval, 25 mobile nodes are randomly placed in a fixed rectangular area of size 2400$m \times$ 1800m with a fixed range of 300m. Each simulation runs for ten thousand steps.

For the Markov Chain prediction model, the number of time steps used for training, n, is 1000. After the training phase, the transition probability matrix derived from training is then used for prediction for the rest of the simulation.

The parameters for Kalman Filter are set as follow: process noise intensity (q) = 0.2; covariance of x (R_x) = 0.2; and covariance of y (R_y) = 0.2. For this analysis, these values are selected by trial and error. A few simulations were executed with a number of different randomly chosen process noise intensity and covariance values, and it is found that relatively good predictions have been made using these chosen values.

3.4.3 Results and Analysis

The results in this section show how well the three selected mobility prediction methods (CVBP, Markov Chain and Kalman Filter) performs against different mobility models.

The results in Figure 3.6 shows the probability of IPLS for GMM using three different prediction methods: deterministic with CVBP method; 2-state Markov Chain link state prediction method; and Kalman Filter prediction method. It illustrates that out of the three prediction methods, Kalman Filter is the best prediction method for GMM. It is expected that the Kalman Filter is a better prediction method compared to the deterministic method for GMM mobility model, because of the stochastic nature of both the GMM and the Kalman Filter. Although the results indicated that the worst prediction method is Markov Chain, this is not to say that Markov Chain is inappropriate for making link state predictions. This is because the Markov Chain link state prediction method that was implemented is a very simple prediction method that predicts the state of the link using only the current link state. Though there may be other more accurate stochastic techniques, such as Markov Models with more states or Hidden Markov Model (HMM), that can be used, they are not implemented for comparison in this study. This is because the Kalman Filter prediction model has shown sufficient improvement in prediction accuracy for the GMM. An exploration of other prediction techniques may be left to future work.

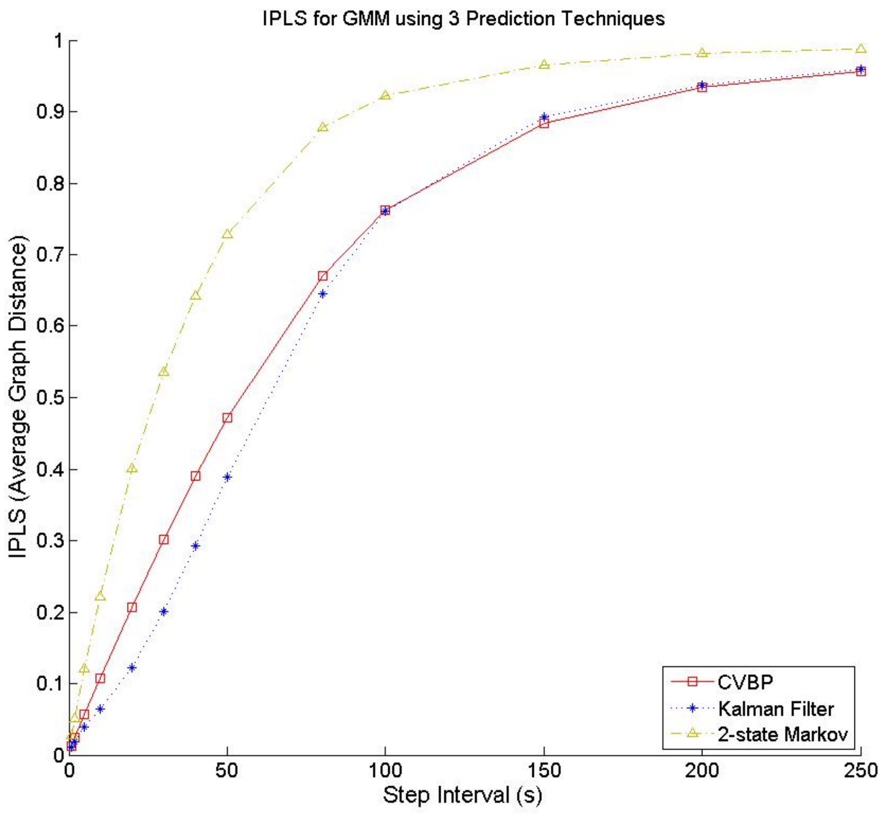


Figure 3.6: *Probability of IPLS for GMM using CVBP, Markov Chain and Kalman Filter prediction methods.*

With no error bars in Figure 3.6, it is difficult to tell if Kalman Filter can really make better predictions compared to CVBP. Thus, a graph with 95% confidence intervals is presented in Figure 3.7. Note that this graph is zoomed in to show the probility of IPLS for GMM motion for step intervals up to 50 seconds. The small confidence intervals on the graph indicates that the confidence levels of the results are high, thus Kalman Filter is a better prediction technique for GMM mobility model.

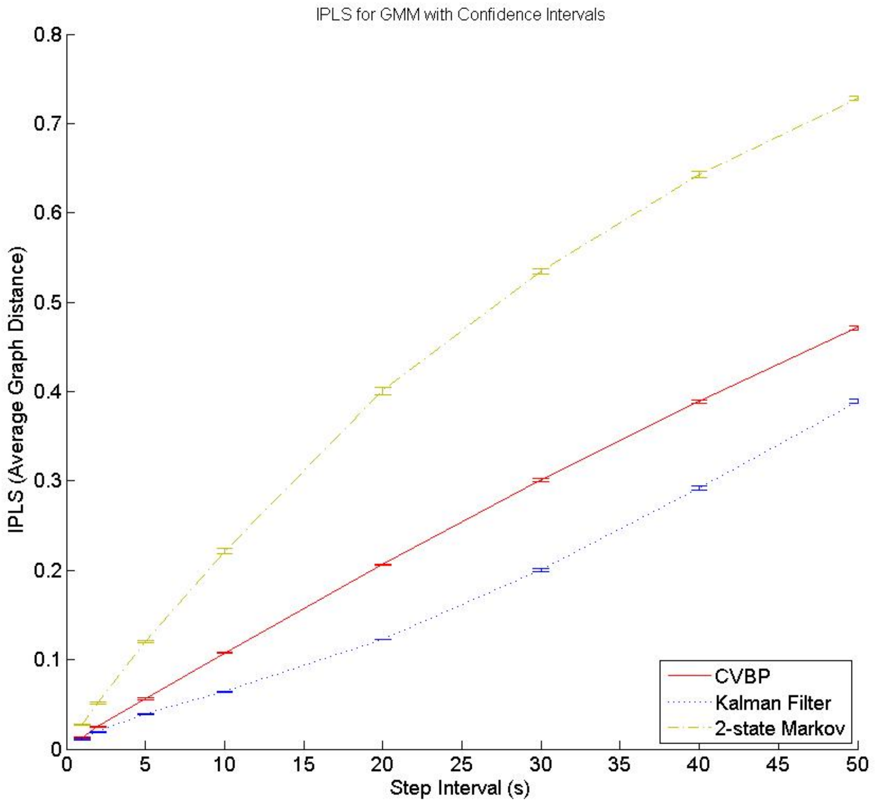


Figure 3.7: *Probability of IPLS for GMM with confidence intervals.*

For a comparison, Figure 3.8 is plotted to show the probability of IPLS for RWM mobility model using deterministic with CVBP method, Markov Chain link state prediction method and Kalman Filter prediction method. This graph indicates that the CVBP method is a better prediction method for RWM, followed by the Kalman Filter prediction method. This is because the nodes in RWM always move at a constant velocity for a selected period of time, the nodes' motion is more deterministic, and hence, deterministic prediction method performs better in this case. Although higher accuracy may be possible with Kalman Filter, it requires tuning the model. This means adaptive prediction method is required to predict different motion.

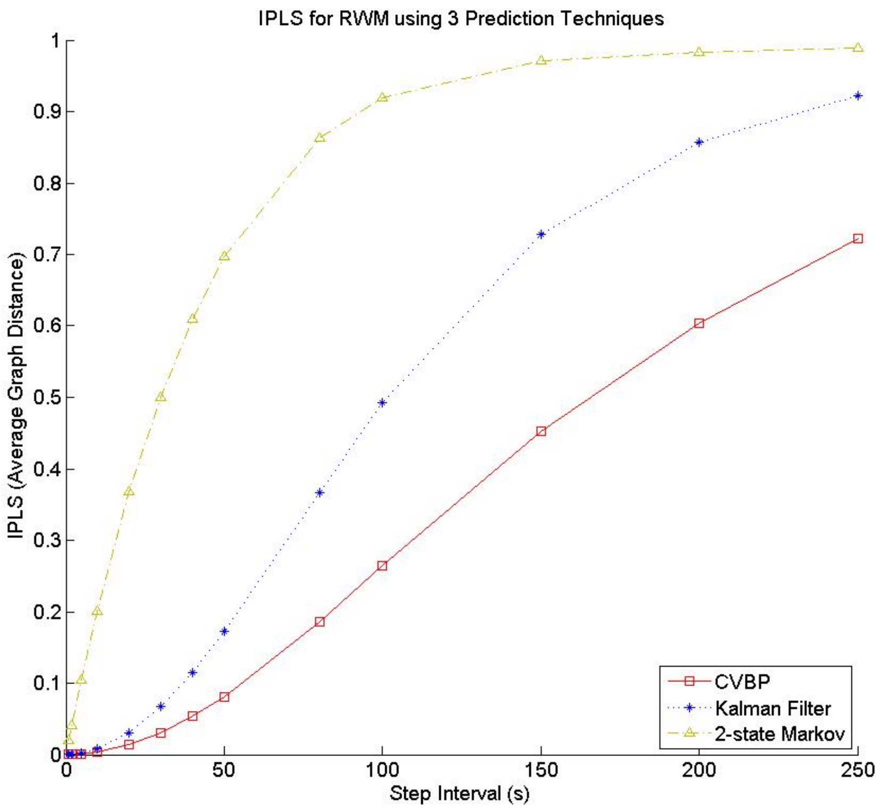


Figure 3.8: *Probability of IPLS for RWM using CVBP, Markov Chain and Kalman Filter prediction methods.*

The results showing the probability of IPLC for GMM using CVBP method, Markov Chain link state prediction method and Kalman Filter prediction method are summarised in Table 3.1. These results indicate that predicting the link change of the nodes using GMM mobility model is not very accurate even with very small time step intervals, and that it is almost impossible to use the Markov Chain method to predict the link change. The reason it is more difficult to predict the link change than to predict the link state is due to the fact that link changes only occur when the nodes are located at a distance close to the maximum transmission range of each other. Thus, any slight randomness in the nodes' movement can reduce the accuracy of the link change prediction.

Step Interval (s)	1	2	5	10
CVBP	0.2021	0.3333	0.4394	0.4765
Kalman Filter	0.1829	0.2677	0.3360	0.3300
Markov Chain	0.3688	0.5821	0.8484	0.9657

Table 3.1: *Probability of IPLC for GMM using CVBP, Markov Chain and Kalman Filter prediction methods.*

In summary, the results indicate that Kalman Filter is a better prediction method for GMM while CVBP is more appropriate for deterministic mobility models such as RWM. Therefore, in order to make accurate predictions, it is important to select the right prediction method and/or tune the prediction model for the type of motion it will be predicting. However, regardless of which prediction method and mobility model are selected, making accurate predictions for a distanced future is very difficult.

3.5 Conclusion

In this chapter, three mobility prediction evaluation metrics are introduced. It has been shown that the prediction accuracy of a network can be evaluated by using different evaluation metrics. Different metrics are useful in evaluating different aspects of prediction accuracy of a network. The accuracy of the location of the nodes can be determined by the Euclidean Distance metrics; the accuracy of the predicted link states can be evaluated by the IPLS method; and the accuracy of the link changes can be evaluated by the IPLC metric.

This chapter also showed that different combinations of mobility models and prediction methods used can affect the accuracy of the predictions. GMMs are more difficult to predict compared to RWMs and RWP models, thus a Kalman Filter is a better prediction method for a GMM, while the CVBP method is a better prediction method for a RWM. The results from the simulations indicate that it is difficult to find a prediction method that can make accurate mobility predictions for many different mobility models. Therefore, the prediction method needs to be adaptive to the mobility models in order to improve the accuracy of the prediction.

Rather than looking more closely into ways to improve the accuracy of prediction to enhance routing performance, the author believes that it would be useful to explore the possible improvements in routing performance when perfect prediction is used for routing. Hence, the effects of perfect prediction has on routing performance are discussed in Chapter 4.

Literature such as [3] has proposed some routing protocols with prediction with the assumption that the prediction needs to be accurate. However, it may not always be true that accurate predictions is necessary for making routing decisions to improve routing performance. This may depend on the routing protocol and the way the prediction information is used. Thus, it is important to explore the situations and the level of accuracy that maybe required to make informed routing decisions in order to improve routing performance in Chapter 5.

Chapter 4

AODV with Perfect Prediction

4.1 Introduction

A number of research studies that have proposed to use prediction to enhance routing performance have been discussed Chapter 2. However, many studies have shown little improvements in routing performance with prediction being used [4, 70, 78]. It is unclear whether this lack of improvement was due to the accuracy of the prediction, the MANET routing protocol used, the mobility of the nodes, or the relevance of statistics studied. It is anticipated that by studying the improvements in routing performance in terms of route lifetime, packet loss and end-to-end delays between the routing protocol without prediction and the routing protocol with perfect prediction, one can identify more clearly the advantages and disadvantages of using prediction, and understand how predictive routing can be improved.

In order to accomplish this, it is assumed that the link duration can be predicted accurately during the route setup stage. In this chapter, a variation of the standard AODV routing protocol, AODV with Perfect Prediction (AODV-PP), has been created to allow the predicted route duration to be broadcast during the route setup stage. This enables the route with the longest lifetime to be selected, which enables the route to better carry the expected traffic that will use this route. Furthermore, it allows a route re-discovery to take place before one of the links along the route disconnects. Since the link duration is predicted accurately, route prediction updates are not implemented in this protocol. Therefore if this protocol is used with inaccurate link duration predictions, the function of a standard AODV protocol would take place to re-establish a route when a link is broken. With perfect prediction, this should be a less likely event.

In this chapter, comparisons between the performance of the standard AODV and AODV-PP are made using network simulations and analysis, by evaluating the data delivery ratio, the average lifetime of the routes and the average route setup/discovery time. This chapter first gives an overview of the AODV routing protocol in Section 4.2, then Section 4.3 explains how AODV-PP works and its differences to standard AODV. Section 4.4 describes how OPNET[1] is being used in the research, the changes made to OPNET's AODV model to perform standard AODV routing, the modifications made to implement AODV-PP, and the tracking model implemented for making link duration predictions. In Section 4.5, an evaluation is made of the difference between AODV and AODV-PP routing protocols and the advantages and disadvantages of using prediction is discussed.

4.2 AODV Routing Protocol

AODV [16] is selected to be the benchmark routing protocol, as it is one of the most commonly used reactive routing protocols. There is much literature about AODV [16, 104] and it has been implemented in OPNET [105]. This section presents an overview of the AODV routing protocol.

[1]OPNET is a network modelling and simulation tool

4.2.1 Route Discovery

AODV is a reactive distance vector routing protocol. This means that a route will only be discovered when it is needed. When a node has a packet to send and there is no route to the destination node, a route request (RREQ) packet will be broadcast from the source node to search for a route, and this node waits for a route reply (RREP) packet from the destination node for a duration of RING_TRAVERSAL_TIME[2]. Upon receiving the RREQ packet, the "Hop Count" field is incremented. If the hop count is greater than the NET_DIAMETER[3], the RREQ packet is discarded.

When a node receives the first RREQ packet, it checks if it has a valid route to the source. If so, and if the RREQ packet has a higher source sequence number, or has the same source sequence number and a smaller hop count (i.e. shorter path), it updates the routing table (or the "route entry"[4]) with the new routing information from the RREQ packet. Any subsequent RREQ packets that do not meet the criteria above will be discarded.

If the receiver of the RREQ packet is an intermediate node, and its "destination-only" flag is enabled, then broadcast the RREQ packet. Otherwise, a RREP packet can be sent back to the source node. The purpose of the "destination-only" flag is explained in more detail in Section 4.2.4. Alternatively, if the receiver of the RREQ packet is the destination, and the RREQ packet has not been discarded, the destination node sends a RREP packet back to the source through the path that the RREQ packet has taken.

Meanwhile at the source node, if a RREP packet has not been received by the

[2]RING_TRAVERSAL_TIME is how long the source will wait for a RREP after a RREQ packet has been sent. This is calculated using a formula specified in the AODV standard, as shown in Table D.1 of Appendix D.1.

[3]NET_DIAMETER is the maximum number of hops allowed between two nodes in the network. See Table D.1 in Appendix D.1 for the default value specified in the AODV standard.

[4]"Route entry" is a structure in OPNET's AODV model that records the current routing information for a specific destination node. The routing information (variables) of an AODV route entry are listed in Table D.2 of Appendix D.2.

source after timeout, the source will broadcast another RREQ packet with a longer timeout period and a higher Time-To-Live (TTL) value. Note that the TTL value can be used to limit the radius of the route search, which can be used to reduce the protocol's overheads. This process repeats until a maximum number of retries has been reached, in which case, the packet(s) queued to be sent to the destination will be discarded. If a RREP packet has been received by the source, all the queued packets will be sent to the destination using the discovered routes.

4.2.2 Route Maintenance

When a node has one or more active routes, a HELLO packet will be broadcast at every HELLO_INTERVAL[5]. If a node has not received a HELLO packet from its neighbour of an active route for more than ALLOWED_HELLO_LOSS[6] $\times$ HELLO_INTERVAL, the node will consider that the link to this neighbour is lost, so the route to this neighbour will be set to inactive.

4.2.3 Route Repair

When a link breaks in an active route, if "local repair" is enabled, and the destination was less than MAX_REPAIR_TTL[7] hops away, the node would repair the route locally by broadcasting a RREQ for the destination. If data packets arrive during the route discovery period of a "local repair", these packets will be buffered. When the discovery period ends, and no RREP has been received for that destination, the buffered packets will be dropped and

[5]HELLO_INTERVAL is the time interval between two HELLO packets from the same node. See Table D.1 in Appendix D.1 for the default value specified in the AODV standard.

[6]ALLOWED_HELLO_LOSS is a constant value. See Table D.1 in Appendix D.1 for the default value specified in the AODV standard.

[7]MAX_REPAIR_TTL is the maximum number of hops allowed for an intermediate node to repair a link. See Table D.1 in Appendix D.1 for the calculation of the parameter in the AODV standard.

a route error (RERR) message will be broadcast to all the precursors of the destination. Although "local repair" may reduce data packet loss, it may cause the applications to fail, due to the increase in end-to-end delay of the buffered packets, and the increase in the number of hops to the associated destinations.

If "local repair" is disabled, the node will broadcast a RERR message to the precursors of all the destinations using that broken link. Upon receiving a RERR message, the node will set the routes to the unreachable nodes to inactive and broadcast a RERR message to the precursors until there are no more precursors. The inactive route will be removed from the route table if it has been inactive for a DELETE_PERIOD[8]. The down side to this configuration is that all the data packets that are destined for the unreachable nodes will be dropped.

4.2.4 Destination-Only Flag

A "destination-only" flag can be used to ensure that only the destination is allowed to send a RREP packet. Although this setting causes longer route discovery time, it ensures that the shortest path (with the least number of hops) can be found at the time of route discovery. When this flag is disabled, an intermediate node is allowed to send a RREP packet upon receiving a RREQ packet if it has a valid route to the destination.

When the "destination-only" flag is disabled, the route discovery time may be reduced. This is because the RREQ packet does not need to reach the destination if the intermediate node has a route to the destination. However, there are a few down sides to this:

1. One problem is that it limits the chance of finding the best possible route;
2. Another problem is that it can cause more route changes due to the possibility of receiving multiple RREPs when there are more than one intermediate nodes having a route to the destination; and

[8]DELETE_PERIOD is the maximum time that allows for a node to keep an inactive route to the destination. See Table D.1 in Appendix D.1 to see how this parameter is obtained.

3. The third problem is that only the route discovered by the destination would be used if the RREQ packet can reach the destination, regardless of whether or not the intermediate nodes have a route to the destination. This is because the sequence number is incremented only when the RREQ reaches the destination, and only the RREP with the highest sequence number will be considered.

However, the option of disabling the “destination-only” flag is not functioning properly in OPNET (see Section 4.4.4.2 for an explanation), therefore this flag is enabled for all simulations.

4.3 AODV with Perfect Prediction

AODV with Perfect Prediction (AODV-PP) has been designed and implemented for this research in order to make comparisons with standard AODV. It has been implemented in a simulated environment where the link durations of all the links in the network are known, i.e. with the assumption that “Perfect Prediction” can be achieved. (It has also been designed so that routing without perfect prediction can be easily implemented as well - this is described in Chapter 5.) Using OPNET simulations, ”Perfect Prediction” can be obtained when the paths of all the nodes in the network are predetermined and the transmission range is constant. The prime purpose of using AODV-PP is to enable AODV to select routes based on the duration or the stability of the routes while searching for a route to the destination, with the assumption that the initial route durations estimated were accurate, so that no duration updates are needed. Therefore, it is designed to operate only when the route duration predictions are accurate. With this assumption, the HELLO packets that were required for AODV can be eliminated. AODV-PP is different to AODV in a number of ways as described in this section.

4.3.1 Route Discovery

As with AODV, route discovery is only required when the node needs to send to the destination, and no route is available. When a data packet arrives from the higher layer of the Open Systems Interconnection (OSI) protocol stack for an unknown or an inactive destination, a RREQ packet will be broadcast to search for the destination, similar to AODV. However in AODV-PP, the RREQ packet requires an additional field, called “Pathchange Time”, to keep track of the estimated path-change time of the return route. This enables the intermediate and destination nodes that have received the RREQ packet to select the route with the longest route lifetime, and to establish a route back to the source node with an estimated route expiry time set to the estimated path-change time. The format of the RREQ packet structure for AODV-PP is illustrated in Appendix B.1.1.

When a RREQ packet is received by a neighbouring node, the following RREQ packet fields are updated:

1. **Hop Count** - This field is initiated to zero by the source, and is incremented by each receiver of the RREQ packet; and
2. **Pathchange Time** - This field is initially set to infinity by the source, and is updated to the minimum value of the previous link’s estimated link-change time and the “Pathchange Time” field of the received RREQ packet.

The previous hop’s route entry is updated if there is a route entry to the previous hop of the RREQ and the estimated link-change time is longer than the route expiry time of the route entry. Alternatively, add a route entry for the previous hop if there there is no existing route entry. This is to maintain the neighbour’s connectivity. In AODV-PP, the route entry is a structure that records the current and the future (pre-route) routing information to a specific node. Note that if the current routing information is valid, all the new routing information updates will be stored temporarily as “pre-route” information until a RREP packet is received (for intermediate nodes) or until it is time to send a RREP (for destination nodes). The routing information of a route entry is listed in Tables D.2 and D.3 of

Appendix D.2. (The variables in Table D.3 are added to AODV for implementing AODV-PP.)

The node will discard the RREQ packet if one of the following conditions is true:

1. If the new predicted path-change time is too short (i.e. less than the current time);
2. If the source route entry exists, and the new source sequence number of the received RREQ packet is less than the current and the pre-route sequence number of the route entry; or
3. If the source route entry exists, the new source sequence number is equal to the pre-route sequence number of the route entry, and the new predicted path-change time is less than the pre-route route expiry time.

If the RREQ packet is the first RREQ received with the same source and destination node, the node does one of the following tasks:

1. It creates a new route entry for the source if no source route entry exists; or
2. It updates the source route entry with the information from the received RREQ packet if one of the following conditions is true:
   (a) The source route entry is not valid; or
   (b) The new source sequence number is greater than the pre-route sequence number.

Alternatively, if a subsequent RREQ packet is received, the node updates the source route entry if one of the following conditions is true.

1. The source route entry is not valid;
2. The new predicted path-change time is longer than the pre-route route expiry time; or
3. The new source sequence number is equal to the pre-route sequence number, and the new predicted path-change time is greater than the pre-route route expiry time.

If the node is not the destination of the RREQ packet, the node will broadcast the

RREQ packet, and set a pre_route_expiry_timer[9] to end the pre-route session if no RREP is received before timeout. Otherwise if the node is the destination, it waits for a period of time that is set according to the TTL value in the IP header that encapsulates the received RREQ packet, before sending a RREP packet at timeout set by the rrep_send_timer[10]. The pre_route_expiry_timer, $t_{PreRexp}$, and the rrep_send_timer, $t_{RrepSend}$, can be computed using Equation 4.1 and 4.2 respectively.

$$t_{PreRexp} = t_0 + 2.0 \times t_{NT} \times (N_{TTL} - 1 + B_{timeout}) \tag{4.1}$$

$$t_{RrepSend} = t_0 + t_{NT} \times N_{TTL} \tag{4.2}$$

where t_0 is the time of the first RREQ arrival; t_{NT} is the NODE_TRAVERSAL_TIME[11]; N_{TTL} is the TTL[12] value; and $B_{timeout}$ is the TIMEOUT_BUFFER[13].

The rrep_send_timer, $t_{RrepSend}$, is set by the destination node to wait for any subsequent RREQs to allow the destination to choose the path with the longest route lifetime and send a single RREP packet. The packet structure of the RREP packet for AODV-PP is illustrated in Appendix B.1.2. The advantages of using the rrep_send_timer is to reduce the number of RREPs sent, and can prevent multiple RREPs to be sent through different paths, which can minimise the number of route changes with very short route duration and can reduce the routing overhead. The drawback is that this can induce extra delay in route discovery. However with AODV-PP, a route can be re-established before any

[9]The pre_route_expiry_timer is the time for the destination node to send a RREP packet, or for the source and the intermediate nodes to end the pre-route state. This timer is added to the route entry to implement AODV-PP, and is listed in Table D.3 of Appendix D.2.

[10]rrep_send_timer is one of the new variables added to the route entry to implement AODV-PP. See Table D.3 in Appendix D.2.

[11]NODE_TRAVERSAL_TIME is the estimate time for a node to transmit a packet to another node. See Table D.1 in Appendix D.1 for the calculation of this parameter.

[12]TTL is an integer that limits the number of hops of the RREQ packet.

[13]TIMEOUT_BUFFER is an integer used in AODV to allow extra time for the RING_TRAVERSAL_TIME to end. See Table D.1 in Appendix D.1 for the default value specified in the AODV standard.

link becomes unavailable[14], hence the problem with delay in route discovery may only be a once-off problem for each source-destination pair.

The difference between the RREP packet in AODV and AODV-PP is that a "Pathchange Time" field is added to the RREP packet to record the estimated path-change time from the destination back to the source. There are two reasons for using this field:

1. One reason is so that the intermediate node will know the route expiry time between itself and the destination, rather than the route expiry time between the source and the destination; and
2. The other reason is to ensure that the source node is getting the latest route expiry time. Although this is not necessary for AODV-PP, this field is useful for routing with inaccurate predictions.

In AODV-PP, the "Lifetime" and the "Pathchange Time" fields of the RREP packet are initially set to the route expiry time of the route to the source and infinity respectively by the destination node. When a node receives a RREP packet, the "Pathchange Time" field is updated to the minimum of the estimated link-change time of the last hop to the destination node and the existing "Pathchange Time" field of the RREP packet.

If the receiver node of the RREP has no route entry to the destination, it sets up a new route entry with the route expiry time set to the new path-change time. Alternatively, a route entry to the destination exists, then update the route entry according to the received RREP packet if at least one of the following cases is true:

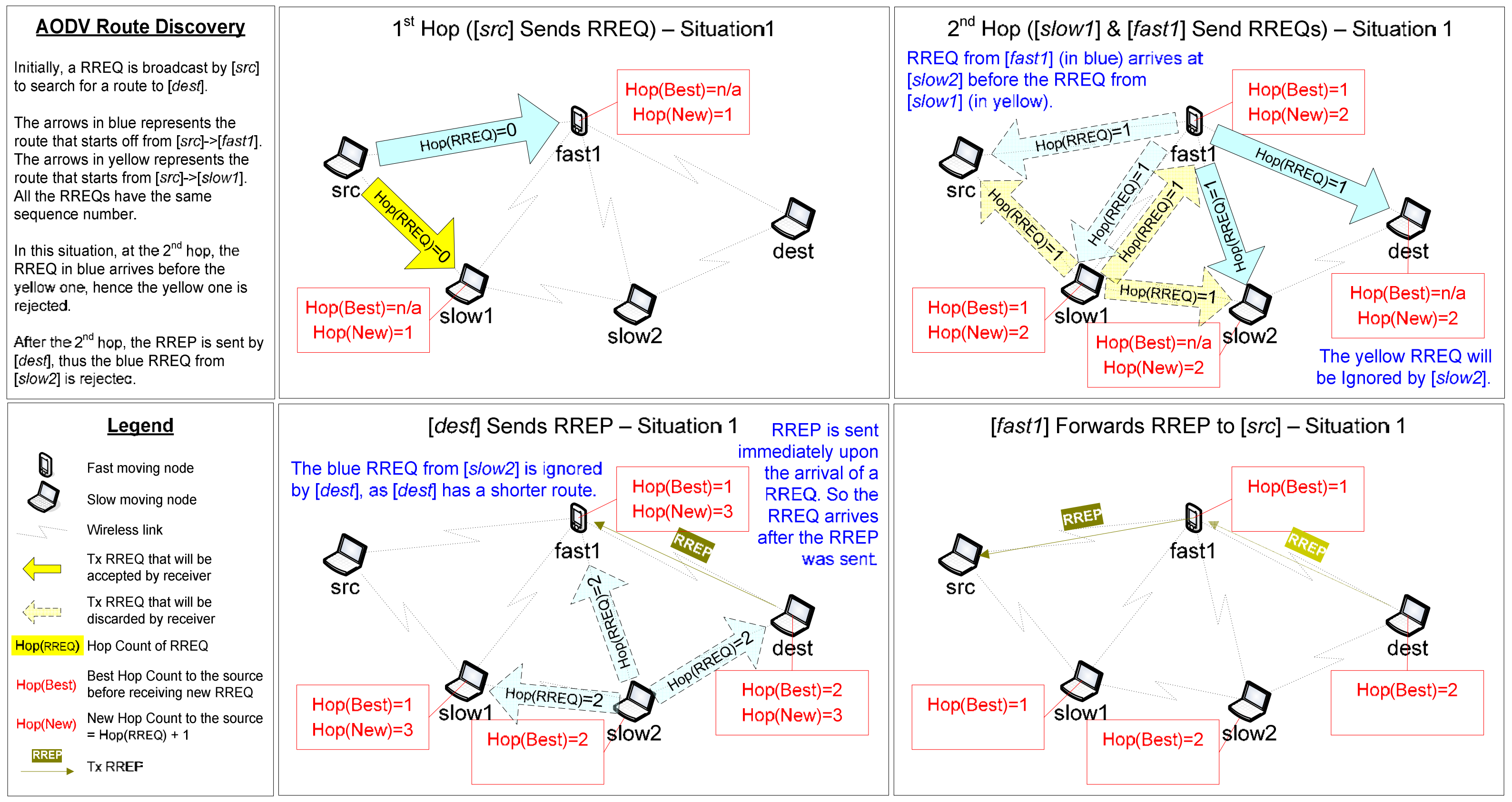
1. If the route entry is invalid;
2. If the destination sequence number of the RREP packet is greater than the sequence number of the route entry; or
3. If the sequence numbers are equal and the updated "Pathchange Time" field of the received RREP packet is greater than the existing route expiry time in the route entry.

[14]Further information about route re-establishment can be found in Section 4.3.2

Once the route entry is updated, if the node that has received the RREP is the source, it starts sending the queued data packets. If not, the node checks if it has a route entry to forward to the source. If the route entry does not exist, then the RREP packet is dropped. Otherwise, if the pre_route_expiry_timer has not expired, then update the route entry with the latest pre-route information that was obtained from the RREQ. The node then forward the RREP packet to the next hop of the forward route entry if the current node is not the source.

To illustrate the differences between the route discovery process of AODV and AODV-PP, some state diagrams are presented in Figures 4.1 to 4.4. Figures 4.1 and 4.2 show the route discovery process for AODV in two different situations over the same network scenario. The other two figures show two different situations for AODV-PP. The difference in the two situations is the order of the RREQ packet received by the receiver. The order of the arrival of the RREQ packets at the intermediate nodes and the destination may affect the route being selected. For all the diagrams, the coloured arrows represent a route discovery through different paths. The sharp brighter coloured arrows represent the propagating RREQs that will be accepted by the receivers, and the lighter coloured arrows are the ones that will be rejected.

The route discovery processes in these diagrams are indicating that the route discovery process of both AODV and AODV-PP begins by broadcasting a RREQ in the same way. A RREQ packet is broadcast by the $[src]$ node to search for the $[dest]$ node, and is received by the neighbouring nodes, $[fast1]$ and $[slow1]$). Each receiver of the RREQ packet will process the received packet, and re-broadcast it to its neighbours. However, for the 2^{nd} hop shown in Figures 4.1 and 4.2 show that in AODV, the node, $[slow1]$, would discard the RREQ from the node, $[fast1]$, as there are more hops in the RREQ from $[fast1]$ compared to the previous route found. Whereas the "2^{nd} hop" diagrams from Figures 4.3 and 4.4 indicate that in AODV-PP, $[slow1]$ would accept the new route, and forward $[fast1]$'s RREQ, which has a longer route lifetime. The process continues until the destination is found. The route discovery processes presented


AODV Route Discovery

Initially, a RREQ is broadcast by [*src*] to search for a route to [*dest*].

The arrows in blue represents the route that starts off from [*src*]->[*fast1*]. The arrows in yellow represents the route that starts from [*src*]->[*slow1*]. All the RREQs have the same sequence number.

In this situation, at the 2nd hop, the RREQ in blue arrives before the yellow one, hence the yellow one is rejected.

After the 2nd hop, the RREP is sent by [*dest*], thus the blue RREQ from [*slow2*] is rejected.

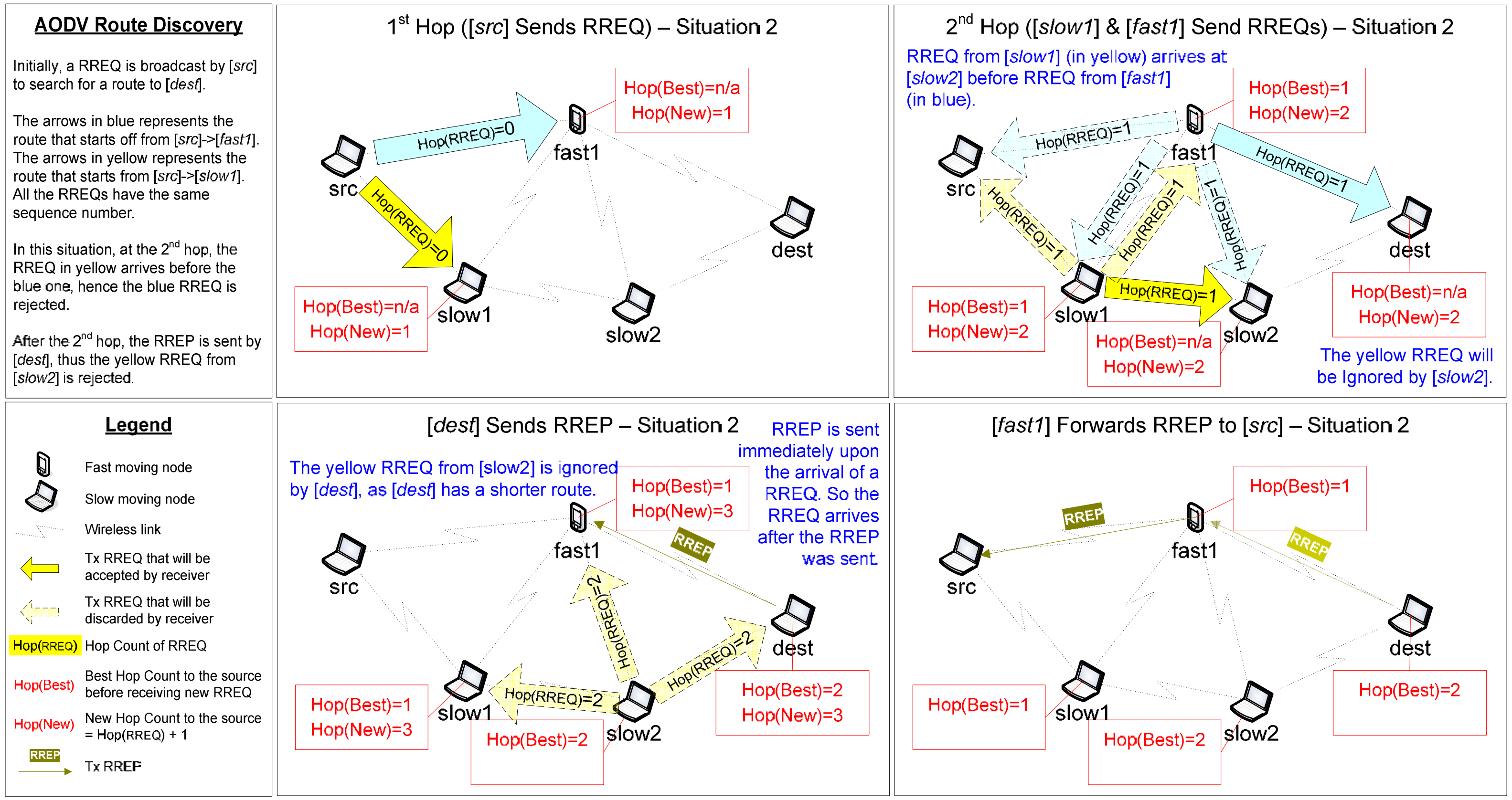
RREQ from [*fast1*] (in blue) arrives at [*slow2*] before the RREQ from [*slow1*] (in yellow).

The yellow RREQ will be Ignored by [*slow2*].

The blue RREQ from [*slow2*] is ignored by [*dest*], as [*dest*] has a shorter route.

RREP is sent immediately upon the arrival of a RREQ. So the RREQ arrives after the RREP was sent.

Figure 4.1: *AODV route discovery example for situation 1.*



AODV Route Discovery

Initially, a RREQ is broadcast by [*src*] to search for a route to [*dest*].

The arrows in blue represents the route that starts off from [*src*]->[*fast1*]. The arrows in yellow represents the route that starts from [*src*]->[*slow1*]. All the RREQs have the same sequence number.

In this situation, at the 2nd hop, the RREQ in yellow arrives before the blue one, hence the blue RREQ is rejected.

After the 2nd hop, the RREP is sent by [*dest*], thus the yellow RREQ from [*slow2*] is rejected.

RREQ from [*slow1*] (in yellow) arrives at [*slow2*] before RREQ from [*fast1*] (in blue).

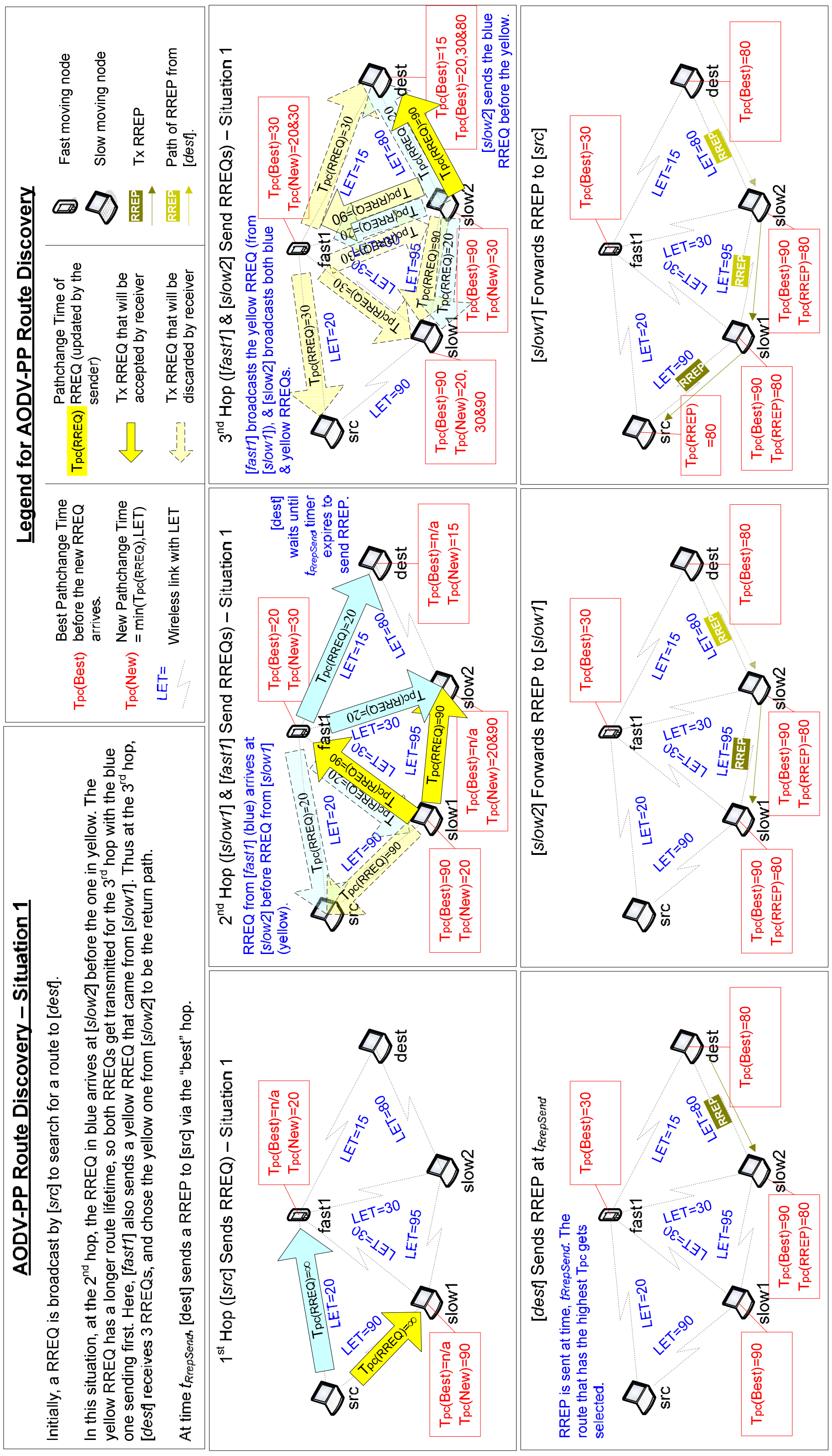
The yellow RREQ will be Ignored by [*slow2*].

The yellow RREQ from [slow2] is ignored by [*dest*], as [*dest*] has a shorter route.

RREP is sent immediately upon the arrival of a RREQ. So the RREQ arrives after the RREP was sent.

1st Hop ([src] Sends RREQ) – Situation 2
2nd Hop ([slow1] & [fast1] Send RREQs) – Situation 2
[dest] Sends RREP – Situation 2
[fast1] Forwards RREP to [src] – Situation 2
src
fast1
slow1
slow2
dest
Hop(RREQ)=0
Hop(RREQ)=1
Hop(RREQ)=2
Hop(Best)=n/a
Hop(New)=1
Hop(Best)=1
Hop(New)=2
Hop(Best)=n/a
Hop(New)=2
Hop(Best)=1
Hop(New)=3
Hop(Best)=2
Hop(New)=3
Hop(Best)=2
Hop(Best)=1
RREP
Legend
Fast moving node
Slow moving node
Wireless link
Tx RREQ that will be accepted by receiver
Tx RREQ that will be discarded by receiver
Hop(RREQ) Hop Count of RREQ
Hop(Best) Best Hop Count to the source before receiving new RREQ
Hop(New) New Hop Count to the source = Hop(RREQ) + 1
RREP Tx RREP

Figure 4.2: *AODV route discovery example for situation 2.*



AODV-PP Route Discovery – Situation 1

Initially, a RREQ is broadcast by [*src*] to search for a route to [*dest*].

In this situation, at the 2nd hop, the RREQ in blue arrives at [*slow2*] before the one in yellow. The yellow RREQ has a longer route lifetime, so both RREQs get transmitted for the 3rd hop with the blue one sending first. Here, [*fast1*] also sends a yellow RREQ that came from [*slow1*]. Thus at the 3rd hop, [*dest*] receives 3 RREQs, and chose the yellow one from [*slow2*] to be the return path.

At time $t_{RrepSend}$, [dest] sends a RREP to [src] via the "best" hop.

Legend for AODV-PP Route Discovery
Tpc(Best) Best Pathchange Time before the new RREQ arrives.
Tpc(New) New Pathchange Time = min(Tpc(RREQ),LET)
LET= Wireless link with LET
Tpc(RREQ) Pathchange Time of RREQ (updated by the sender)
Tx RREQ that will be accepted by receiver
Tx RREQ that will be discarded by receiver
Fast moving node
Slow moving node
RREP Tx RREP
RREP Path of RREP from [dest].
1st Hop ([src] Sends RREQ) – Situation 1
2nd Hop ([slow1] & [fast1] Send RREQs) – Situation 1
RREQ from [fast1] (blue) arrives at [slow2] before RREQ from [slow1] (yellow).
[dest] waits until t_RrepSend timer expires to send RREP.
3rd Hop ([fast1] & [slow2] Send RREQs) – Situation 1
[fast1] broadcasts the yellow RREQ (from [slow1]), & [slow2] broadcasts both blue & yellow RREQs.
[slow2] sends the blue RREQ before the yellow.
[dest] Sends RREP at t_RrepSend
RREP is sent at time, t_RrepSend. The route that has the highest Tpc gets selected.
[slow2] Forwards RREP to [slow1]
[slow1] Forwards RREP to [src]
src
fast1
slow1
slow2
dest
LET=20
LET=90
LET=30
LET=95
LET=15
LET=80
Tpc(RREQ)=∞
Tpc(RREQ)=90
Tpc(RREQ)=20
Tpc(RREQ)=30
Tpc(Best)=n/a Tpc(New)=90
Tpc(Best)=n/a Tpc(New)=20
Tpc(Best)=90 Tpc(New)=20
Tpc(Best)=20 Tpc(New)=30
Tpc(Best)=n/a Tpc(New)=20&90
Tpc(Best)=n/a Tpc(New)=15
Tpc(Best)=90 Tpc(New)=20, 30&90
Tpc(Best)=30 Tpc(New)=20&30
Tpc(Best)=90 Tpc(New)=30
Tpc(Best)=15 Tpc(Best)=20,30&80
Tpc(Best)=90
Tpc(Best)=30
Tpc(Best)=80
Tpc(Best)=90 Tpc(RREP)=80
Tpc(RREP) =80

Figure 4.3: AODV-PP route discovery example for situation 1.

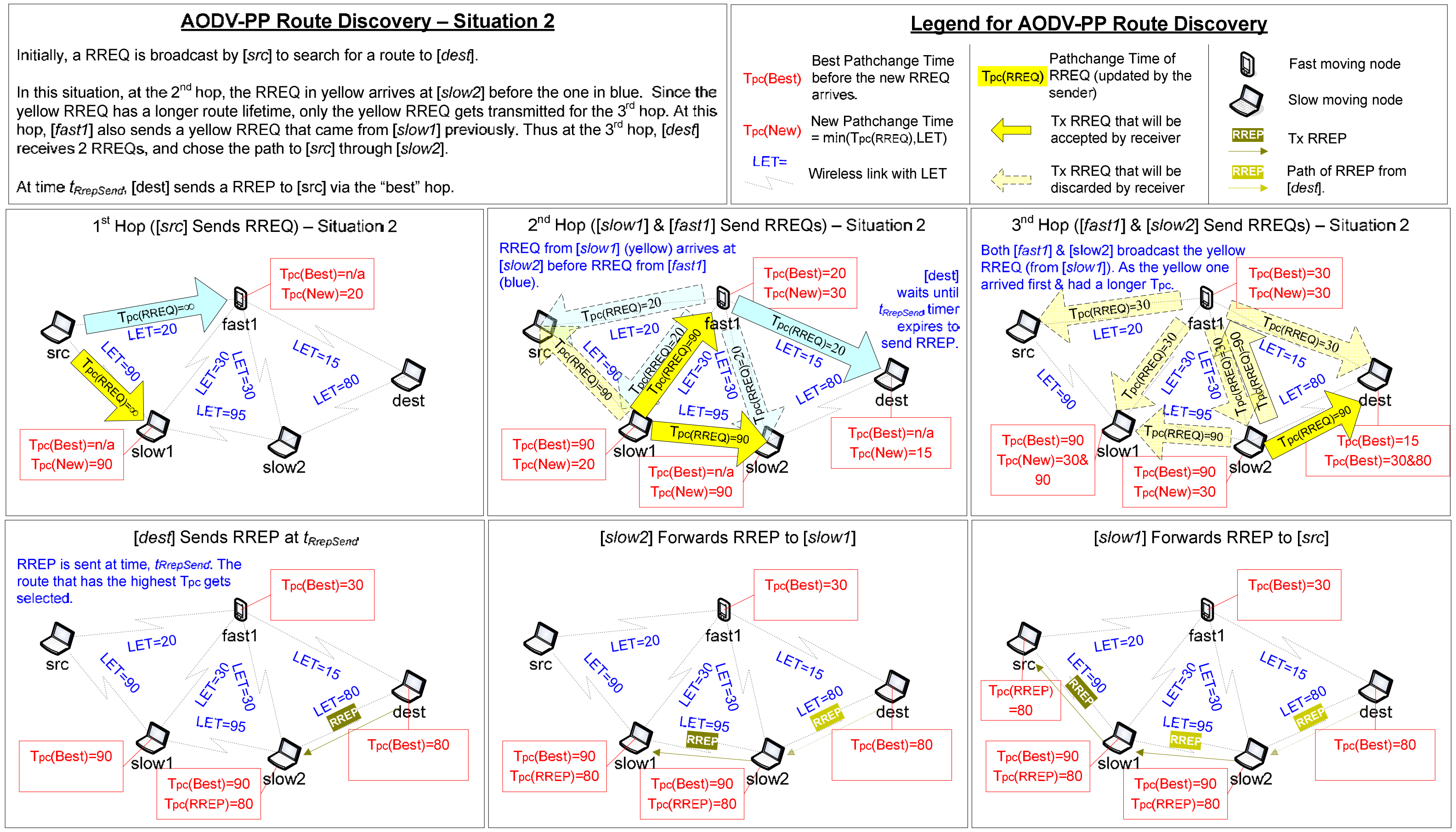

AODV-PP Route Discovery – Situation 2
Initially, a RREQ is broadcast by [src] to search for a route to [dest].
In this situation, at the 2nd hop, the RREQ in yellow arrives at [slow2] before the one in blue. Since the yellow RREQ has a longer route lifetime, only the yellow RREQ gets transmitted for the 3rd hop. At this hop, [fast1] also sends a yellow RREQ that came from [slow1] previously. Thus at the 3rd hop, [dest] receives 2 RREQs, and chose the path to [src] through [slow2].
At time tRrepSend, [dest] sends a RREP to [src] via the "best" hop.
Legend for AODV-PP Route Discovery
Tpc(Best) Best Pathchange Time before the new RREQ arrives.
Tpc(New) New Pathchange Time = min(Tpc(RREQ),LET)
LET= Wireless link with LET
Tpc(RREQ) Pathchange Time of RREQ (updated by the sender)
Tx RREQ that will be accepted by receiver
Tx RREQ that will be discarded by receiver
Fast moving node
Slow moving node
RREP Tx RREP
RREP Path of RREP from [dest].
1st Hop ([src] Sends RREQ) – Situation 2
2nd Hop ([slow1] & [fast1] Send RREQs) – Situation 2
RREQ from [slow1] (yellow) arrives at [slow2] before RREQ from [fast1] (blue).
[dest] waits until tRrepSend timer expires to send RREP.
3rd Hop ([fast1] & [slow2] Send RREQs) – Situation 2
Both [fast1] & [slow2] broadcast the yellow RREQ (from [slow1]). As the yellow one arrived first & had a longer Tpc.
[dest] Sends RREP at tRrepSend
RREP is sent at time, tRrepSend. The route that has the highest Tpc gets selected.
[slow2] Forwards RREP to [slow1]
[slow1] Forwards RREP to [src]
src
fast1
slow1
slow2
dest
LET=20
LET=90
LET=30
LET=15
LET=80
LET=95
Tpc(RREQ)=∞
Tpc(RREQ)=20
Tpc(RREQ)=90
Tpc(RREQ)=30
Tpc(Best)=n/a
Tpc(New)=20
Tpc(New)=90
Tpc(Best)=90
Tpc(New)=15
Tpc(Best)=20
Tpc(New)=30
Tpc(Best)=30
Tpc(New)=30&90
Tpc(Best)=15
Tpc(Best)=30&80
Tpc(Best)=80
Tpc(RREP)=80

Figure 4.4: *AODV-PP route discovery example for situation 2.*

in the state diagrams for AODV indicate that AODV is able to find its “best”[15] route with the least number of hops, but it may have a short lifetime. And the route discovery processes presented in the diagrams for AODV-PP illustrate that AODV-PP selects the route with the longest lifetime, but with more number of hops.

These state diagrams have shown how AODV-PP selects the most long-lived route, while AODV selects the path with the least hops, which may have a shorter route lifetime. Therefore, if there are links with different longevity in a mobile network, AODV may have a higher tendency to find routes with shorter lifetimes, while AODV-PP will be able to find the ones that are the most long lasting.

4.3.2 Route Re-establishment

In AODV-PP, the source node is required to find a new route before a link becomes unavailable. To achieve this, the source estimates the time that it may take to re-route before the route expiry time, and sets the pre_route_start_timer[16], $t_{RrepSend}$. When the pre_route_start_timer is up, a Pre-Route RREQ (PreRREQ) packet is broadcast.

The pre_route_start_timer, $t_{PreRREQ}$, is defined in Equation 4.3

$$t_{PreRREQ} = t_{Rexp} - t_{MaxRingTrav} \tag{4.3}$$

where t_{Rexp} is the current route expiry time of the route entry; and $t_{MaxRingTrav}$ is the MAX_RING_TRAVERSAL_TIME that is defined as:

$$t_{MaxRingTrav} = 2.0 \times t_{NT} \times (N_d + B_{timeout}) \tag{4.4}$$

[15]The meaning of “best” is protocol dependent.

[16]The pre_route_start_timer is the time to send Pre-Route RREQ (PreRREQ), and it is one of the variables added to the route entry to implement AODV-PP. This timer is listed in Table D.3 of Appendix D.2.

where t_{NT} is the NODE_TRAVERSAL_TIME; N_d is the NET_DIAMETER; and $B_{timeout}$ is the TIMEOUT_BUFFER. These three parameters used here are the default values specified in the AODV standard [16], and are replicated in Table D.1 of Appendix D.1.

Similar to what happens when a node receives a RREQ, the nodes that have received the PreRREQ packet and have a valid route to the source will keep a record of the new best possible return route to the source for update after a timeout, while it continues to use the old route to forward the data packets. If the received PreRREQ has a better route duration, it will update the “Pathchange Time” and the “Hop Count” fields of the PreRREQ and re-broadcast the PreRREQ packet. At intermediate nodes that have broadcast the PreRREQ, the pre_route_expiry_timer is set. If this timer expires before the receipt of a RREP, the new pre-route record will be discarded.

Upon receiving a PreRREQ at the destination, and if it is the first PreRREQ received from the source, the pre-route record of the route entry for the source will be updated, and wait until the rrep_send_timer is up. Between the current time and the end of the rrep_send_timer, if another PreRREQ from the same source arrives, and has a longer route expiry time, the pre-route route entry will be updated with this new route. When the rrep_send_timer expires, the destination will send a RREP back to the source, as explained in Section 4.3.1.

4.3.3 Route Maintenance

As previously mentioned, route maintenance is not needed for AODV-PP, as the prediction is assumed to have 100% accuracy. However, to ensure that this protocol will not fail with imperfect prediction, route maintenance is implemented. This is achieved by estimating the link-change time upon receiving a data packet. This enables a node to update the connectivity status (and the route expiry time) of its neighbouring nodes, so that if a link is disconnected, the routes that ultilse the disconnected link can be removed immediately. (As opposed to AODV, a node does not know a neighbouring node is disconnected until some time after the link becomes unavilable.)

If a data packet is received for a destination that the node has no valid route entry to, a RERR packet is sent or broadcast to the precursor nodes. Upon receiving a RERR packet, the node will create and forward a new RERR packet containing any destinations in the node's routing table with one or more precursors that are listed as unreachable in the received RERR packet. If the node is the source, a subsequent RREQ is broadcast for route re-establishment.

4.4 Network Modelling and Implementations

As mentioned previously, network modelling and simulations are used to compare and analyse AODV and AODV with Perfect Prediction. This can be done using existing network modelling and simulations tools. Products such as NS-2 [106], OPNET [105], Qualnet [107] and OMNeT++ [108], are some of the most commonly used network simulators [109–112].

OPNET is chosen to be the network simulator used for this study, as it is a proprietary network modelling and simulation tool, and it is one of the most widely used and well developed commercial communication networks tools available, with reputation for model fidelity, and has client support. As such, it has a large selection of built-in models for the OSI protocol stack from physical layer up to application layer, so a MANET simulation with standard protocols can be set up and executed very quickly. On the other hand, it is highly flexible, allowing the users to modify the built-in models and to implement new protocols and models using C/C++ programming language. These models can be very simple with low fidelity, or very complex with high fidelity. This enables users to build networks with simple but less realistic nodes, or networks with detail and more realistic nodes with the full OSI stack.

To accomplish this study, modifications have been made to a number of existing models, and new models have been added to implement prediction in AODV. A tracking model is implemented and incorporated to each mobile node in OPNET to obtain and predict the locations of the nodes, and to predict the time when link's connectivity will be lost. The relevant OPNET issues with the AODV model have been addressed. OPNET's AODV routing protocol model has been modified to access and utilise the prediction information for routing using AODV-PP. This was a major modification of OPNET's AODV protocol and required extensive programming in the OPNET environment.

This section of the chapter explains the models used for this study, and the issues in the existing models, and the modifications made to the OPNET models.

4.4.1 MANET Model

In OPNET, a MANET is modelled by placing some wireless nodes onto a rectangular field, as shown in Figure 4.5. Each of these wireless nodes is a node model. The white lines represent the motion of the nodes. This means that the nodes that have no white lines attached to it indicate that those nodes are stationary. Notice that there is a node that looks different to the others labelled "SmartMAC". This node is a central configuration tool that is used to configure the "Smart MAC" model's attributes that are common to all the nodes in the network to perform physical layer functions which is to be explained in Section 4.4.3.

Figure 4.6 shows a node model that consists of various process models (the square blocks) that are used to model each node of the MANET. This node model consists of a physical layer, a Medium Access Control (MAC) layer, an Address Resolution Protocol (ARP) layer, a routing layer, a transport layer and an application layer. The physical layer consists of a transceiver port and a receiver port. The MAC layer uses the "Smart MAC" model. The ARP layer is modelled by the "ARP" model. The routing layer consists of the Internet Protocol

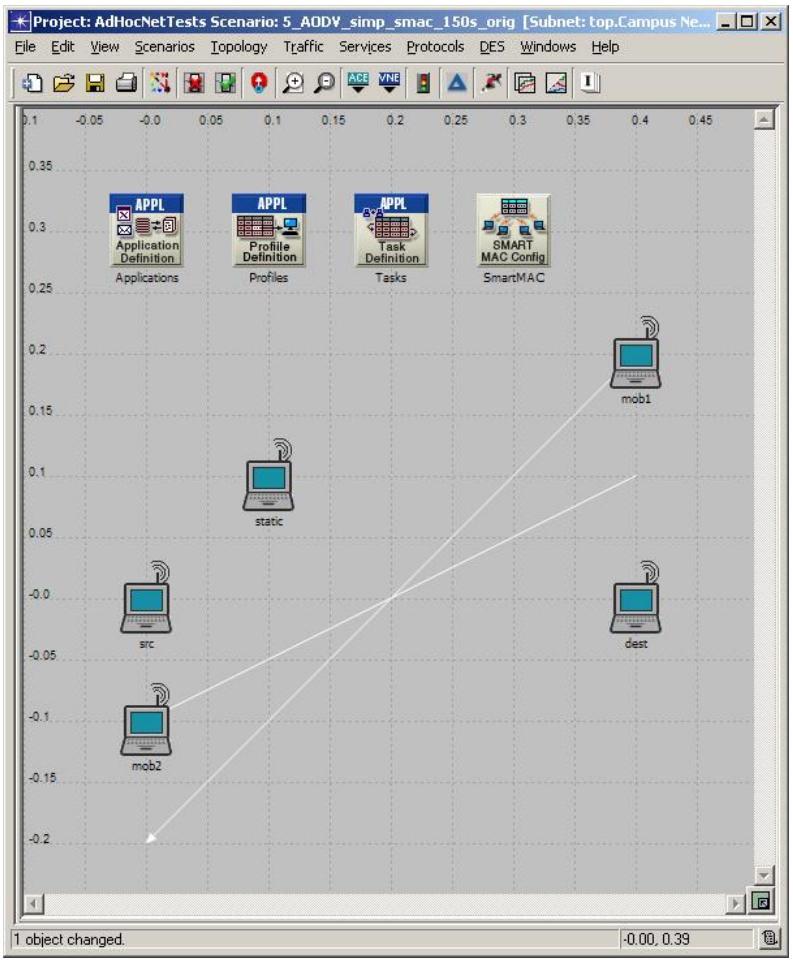

Project: AdHocNetTests Scenario: 5_AODV_simp_smac_150s_orig [Subnet: top.Campus Ne...
File Edit View Scenarios Topology Traffic Services Protocols DES Windows Help
Application Definition
Applications
Profile Definition
Profiles
Task Definition
Tasks
SMART MAC Config
SmartMAC
mob1
static
src
dest
mob2
1 object changed.
-0.00, 0.39

Figure 4.5: *OPNET - A 5-node network model of a MANET.*

(IP) and the AODV/AODV-PP protocols. User Datagram Protocol (UDP) is used by the transport layer. The application layer is implemented by a source generator, “traf_src”.

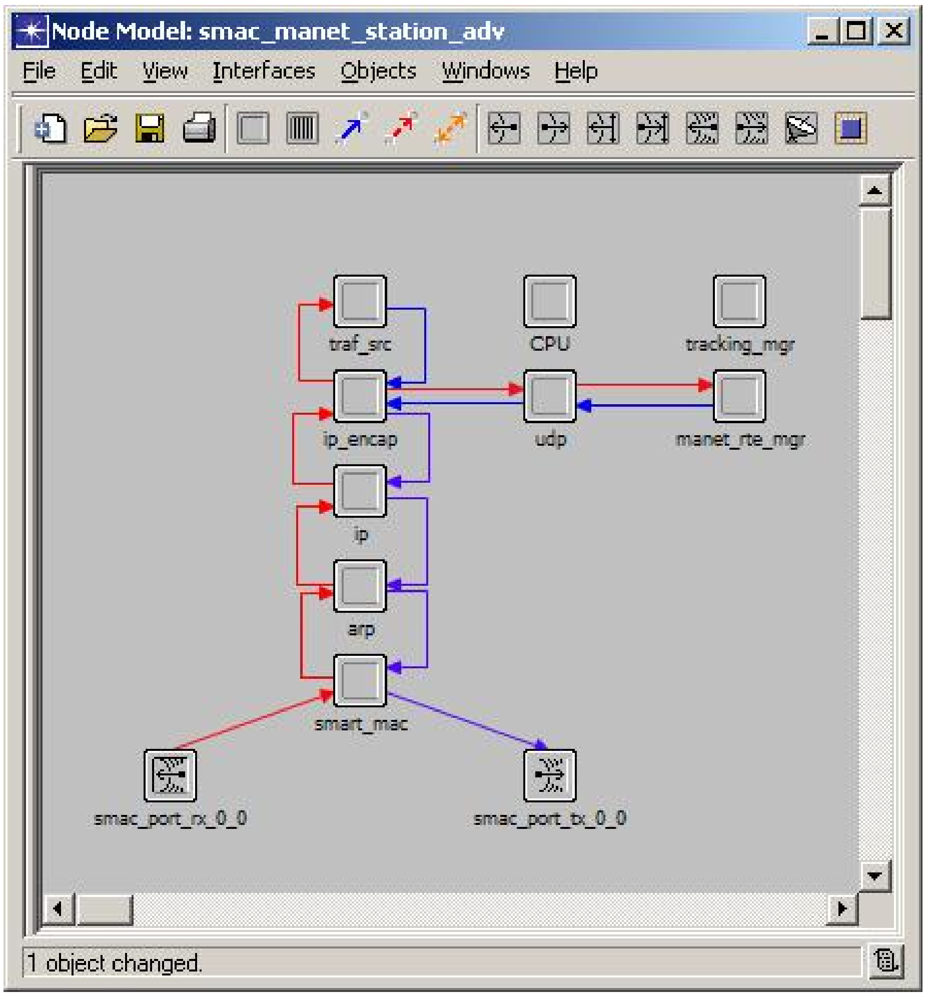


Figure 4.6: *OPNET - A sample of a node model.*

The functions of a node can be implemented by one or more process models. A process model is a state diagram which can be programmed in C/C++ to imitate any protocol in the OSI layer stack or other processes to send, receive and process packets to implement the function of a particular protocol.

4.4.2 Tracking Process Model

OPNET built-in models do not come with any tracking functions to make predictions. Hence, a tracking process model, called "tracking_mgr" (see Figure 4.7), has been designed and written in OPNET to make predictions. This process model is added to all the nodes in the network (i.e. the node model). The "tracking_mgr" model is designed to make perfect link duration predictions by calculating the exact times when the link changes will take place by using the full knowledge of the location and motion (i.e. velocity) of all the nodes in the network. Although the "tracking_mgr" model is designed to make perfect link duration prediction, it is implemented in a way so that other prediction methods may be added in the future for future research.

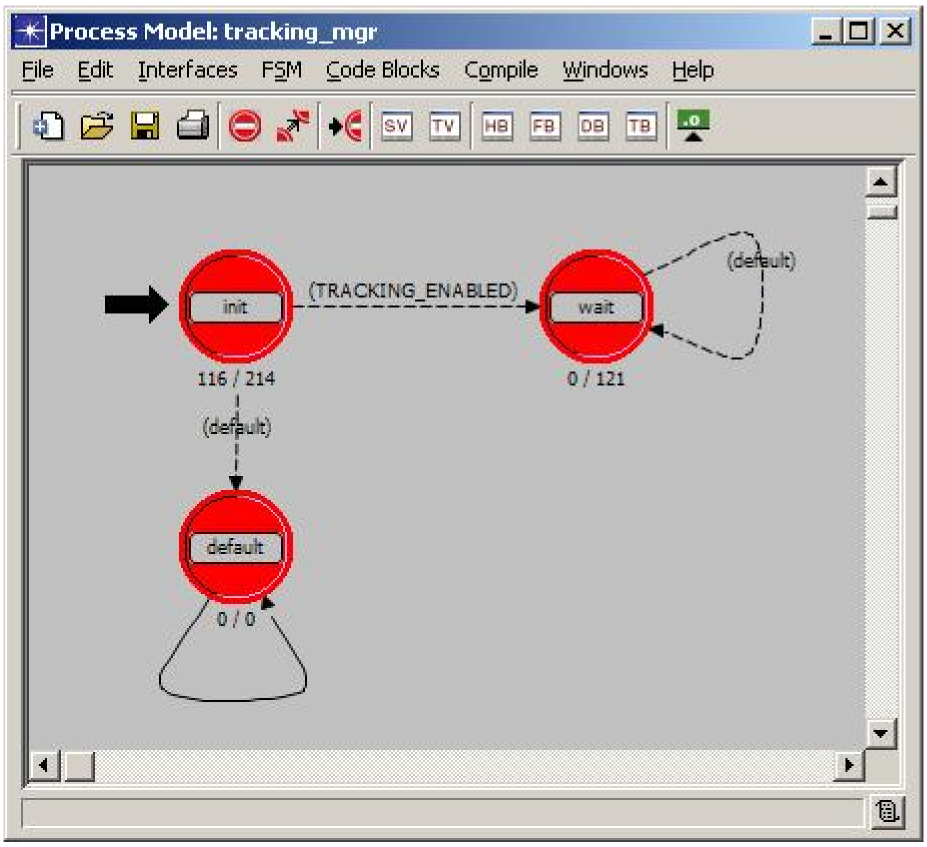


Figure 4.7: *OPNET - The "tracking" process model.*

The “tracking_mgr” model has the following attributes that can be easily set or modified by the user from the network model:

1. **Tracking** - This determines whether tracking is enabled or disabled. This can be set to ”disable” if prediction is not needed for the routing protocol;
2. **Delay** - It specifies the delay between the time its own location is sent and the time location information of other nodes are received;
3. **Sampling Interval** - The sampling interval in seconds is the time gap between sending its own positions;
4. **History Size** - The number of previous location information records that a node will use for making future predictions (this is not used for perfect prediction);
5. **Prediction Method** - When there are other prediction methods implemented, a prediction method can be selected; and
6. **Prediction Parameters** - This includes the Future Size indicating the number of future predictions that will be made, Connect Time Enable indicates whether the link connection time will be predicted, Link State Enable indicates whether the link states of the links will be predicted, and SNR Enable indicates whether the Signal-to-Noise Ratio (SNR) will be predicted.

Initially, at the beginning of the simulation, this model computes the link-change times between all the nodes in the network. This will help analyse the accuracy of prediction when different prediction methods are implemented.

During a simulation, this “tracking_mgr” process is interrupted periodically at a time given in the “sampling interval” attribute to retrieve the location of its own node. The node then sends its own location information to all other nodes in the network (this is done without using the wireless medium). After a “delay” period, the other nodes will obtain the locations of all the other nodes. Each node will store a series of all the nodes’ locations limited by the “history size” specified by the user.

4.4.3 Physical and MAC Layer Models

The physical and the Medium Access Control (MAC) layers control the point-to-point transmission between any two nodes of the network. OPNET provides a number of different built-in models for the MAC layer such as the “Wireless LAN” model, the “WiMAX” model, the “TDMA” model, and the “Smart MAC” model. The “Wireless LAN”, “WiMAX” and the “TDMA” models imitate the IEEE 802.11 WiFi standard, the WiMAX standard and Time Division Multiple Access (TDMA) standard respectively, where as “Smart MAC” is a model that has no protocol, it simply calculates the validity of a point-to-point link by distance calculations without any real MAC protocol.

For the purpose of this study, the physical and the MAC layer are simplified by using “Smart MAC”, as shown in Figure 4.8. Using this “Smart MAC” model, the nodes are able to transmit packets to the receiver at anytime without collisions as long as the two nodes are in range, which is determined by the distance between the two nodes, and that the receiver will always be able to receive the packet transmitted by the sender.

This process model has been modified so that it provides an option to the user to choose to transfer neighbours’ connectivity and location information to AODV-PP with every packet transfer (even when there is no traffic sent to a specific node). This modification can provide more location information data to AODV-PP to make more accurate predictions. However, there is a constant stream of traffic generated by the source nodes in the simulation scenarios designed for this study, so this option is not activated.

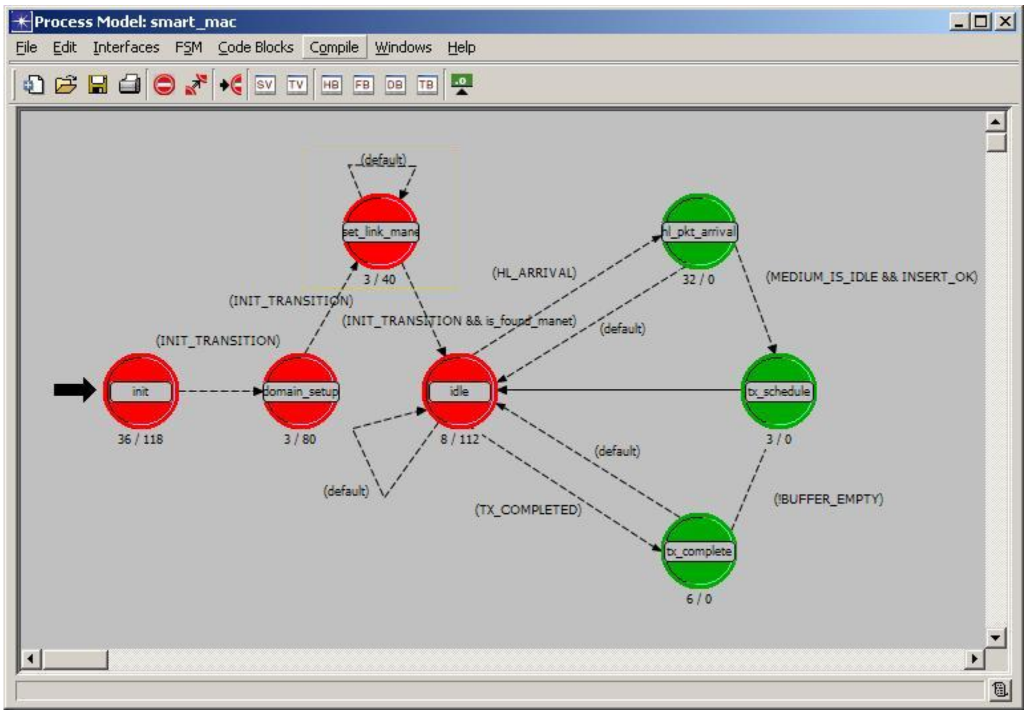


Figure 4.8: *OPNET - The "Smart MAC" process model.*

4.4.4 Routing Layer Models

AODV routing protocol is the focus of this thesis, thus it is important to ensure that it is functioning correctly. In order to verify the AODV model, small network simulations were designed to test and verify the AODV routing protocol. The tests have shown that there are some issues with the current OPNET AODV simulation that must be corrected before the AODV-PP can be implemented. These issues and the solutions have been forwarded to the OPNET support team. They have noted and have agreed on the issues on the OPNET Support Center website (Software Report ID 153334) [113], and they have agreed on some of the solutions. These problems are explained later in this section, and the solutions are provided in Appendix A. In the routing layer, simulations have shown that the performance of AODV in OPNET has a very high packet loss ratio. Furthermore, the default setting of enabling the "local repair" function and the disabling the "destination-only" flag have caused some issues in some simulation scenarios that have been tested.

Identifying the problems in a routing protocol is not an easy task. This is because routing protocol problems are not simple programming bugs that will give errors while executing. In many cases, the protocol appears to work fine in simulations, i.e. routes can be found and packets can be sent to the destination, but poor routing performance may be observed. However, because poor routing performance in MANET can be caused by many reasons, such as the mobility of the nodes, the bit-error-rate, the signal-to-noise ratio, packet collision, contention, jitter etc. It is difficult to know what causes the routing performance to drop. Therefore, to eliminate these ambiguities, a "Smart MAC" model (as mentioned in Section 4.4.3) with no errors, no collisions, no packets drop and no contention has been used, and all jitter delays has been removed from the protocols. Even with these settings simplifying the problem, it is difficult to find the problems in routing when the protocol works, as it can perform the functions of a routing protocol, but only lacks the performance. Furthermore, there are occasions when the problems seem to have gone away after changing some parameters, but some subtle problems reappear after expanding the network size.

In which case, only careful analysis of the results can distinguish the problems in the routing protocol.

Once it is realised that there are problems, identifying what causes the problem is very time consuming, as detailed step by step traces often need to be used.

4.4.4.1 OPNET's AODV - Local Repair Issue

It is found that when the "local repair" option in OPNET is enabled, there are cases when routes may become very unstable. For example, consider the 5-node network scenario as shown in Figure 4.5 on page 79. The 5 nodes are: $[src]$, $[dest]$, $[static]$, $[mob1]$ and $[mob2]$. In this scenario, the $[src]$ node is required to send packets to the $[dest]$ node, with the $[src]$ node being permanently in range with the $[static]$ node. In this network, there is no path available to the destination initially. The path will become available when the mobile nodes, $[mob1]$ and $[mob2]$, move to some positions where they can form a route between the source and the destination.

Initially in this scenario, the $[src]$ node is in range with the $[static]$ and the $[mob2]$ nodes, while the $[dest]$ node is in range with the $[mob1]$ node. At time $150s$, the $[mob1]$ node begins to move towards the southwest direction at a speed of $2m/s$. When the node, $[mob1]$, is in range with the $[static]$ node and later with the $[src]$ node, they are then in range with one another. The node, $[mob1]$, then stops for 44s and starts moving again at a speed of $10m/s$, so it drops out of range from the $[dest]$ node, and then later drops from the $[static]$ node. When the node, $[mob2]$, moves towards the northeast at time $171.44s$ at a speed of $10m/s$, it first becomes in range with the $[dest]$ node, then falls out of range with the $[src]$ node, and later also drops out with the $[static]$ node. Finally the node, $[mob2]$, turns around and moves back towards its original location at a speed of $1m/s$. In this scenario, the "local repair" function fails in the following situation:

1. The first established route to this scenario is: $[src] \rightarrow [static] \rightarrow [mob1] \rightarrow [dest]$.

2. When link $[mob1] \rightarrow [dest]$ breaks, $[dest]$ sets the routes to both $[src]$ and $[mob1]$ as invalid.
3. $[mob1]$ sends a RERR to $[static]$.
4. In the mean time, $[src]$ has a data packet for $[dest]$, so it sends the data packet to $[static]$, as $[src]$ does not yet know the route to $[dest]$ is invalid.
5. $[static]$ receives the RERR from $[mob1]$, and sends a RERR to $[src]$.
6. $[src]$ receives a RERR from $[static]$, and removes its route to $[dest]$.
7. $[static]$ receives a data packet for $[dest]$ from $[src]$, but has no route to $[dest]$, thus it broadcasts a RREQ and queue the data packet.
8. $[mob2]$ receives a RREQ from $[static]$, and broadcasts the RREQ.
9. $[dest]$ receives a RREQ from $[mob2]$ (at this stage, $[dest]$ and $[mob2]$ are in range), and sends a RREP back to $[static]$ via $[mob2]$. Now ($[dest]$ has active routes to both $[static]$ and $[mob2]$).
10. $[mob2]$ receives the RREP from $[dest]$ to $[static]$, and forwards it to $[static]$.
11. $[static]$ receives the RREP from $[dest]$ via $[mob2]$, so $[static]$ now has a path to $[dest]$ (i.e. $[static] \rightarrow [mob2] \rightarrow [dest]$), and sends the queued packet to $[dest]$.
12. $[src]$ has a data packet for $[dest]$, but does not have a route to $[dest]$ (because it had received a RERR from $[static]$ in step 6). So $[src]$ broadcasts a RREQ.
13. $[static]$ receives a RREQ from $[src]$, and has a route to $[dest]$. Thus it sends a RREP to $[src]$.
14. $[mob1]$ also receives the RREQ from $[src]$, but has no route to $[dest]$, so it broadcasts the RREQ.
15. $[src]$ receives the RREP from $[static]$, but it has the same sequence number as the current sequence number, no changes is made to the route to $[dest]$. (Route = $[src] \rightarrow [static] \rightarrow [mob2] \rightarrow [dest]$).
16. $[mob2]$ receives a RREQ from $[mob1]$, and has a route to $[dest]$. It sends a RREP to $[src]$ via $[mob1]$.
17. $[mob1]$ receives the RREP from $[mob2]$, and forwards it to $[src]$.
18. $[src]$ receives the RREP from $[mob1]$, but it has the same destination sequence number as the existing sequence number, no changes is made to the route to $[dest]$ (Route = $[src] \rightarrow [static] \rightarrow [mob2] \rightarrow [dest]$).

19. [*dest*] only knows that it has a route connection with [*static*] and no other nodes, but [*static*] is not the source of the data packets, so the route expiry time for [*static*] is not updated, and thus expires due to not receiving any data packets from [*static*]. As a result, there is no more active route exist in [*dest*], and hence stops broadcasting HELLO packets.
20. [*mob*2] stops receiving HELLO packets from [*dest*], and perceives that its link to [*dest*] has been broken, so it broadcasts a RERR packet.
21. [*src*] has a data packet for [*dest*], so it sends the data packet to [*static*], as [*src*] does not yet know the route to [*dest*] is invalid.
22. [*dest*] receives the RERR from [*mob*2], but has no unreachable destination.
23. [*static*] receives the RERR from [*mob*2], removes the active route to [*dest*], and sends the RERR to [*src*].
24. [*src*] receives the RERR from [*static*], and removes the active route to [*dest*].
25. Repeat from step 7.

The scenario described above shows that with "local repair" enabled, it is possible that this routing protocol can continue to loop continuously with RREQ, RREP and RERR packets if the [*src*] node continues to send a data packet before it has received a RERR from node 3. This is due to the fact that during "local repair", the intermediate node can send a RREQ to the destination, and the destination falsely believes that the intermediate node is the source. Hence, the route expiry time is not updated as it receives a data packet from the [*src*] node. When the only active route at the [*dest*] node expires, it stops sending HELLO messages. Eventually, its neighbours disconnect because no HELLO messages have been received, and activates a RERR to be sent. If the data packet from source comes at the right time, it can trigger another "local repair". This problem exists in the fundamental OPNET AODV model.

According to the scenario above, there seems to be a number of problems with the "local repair" function. These problems include:

1. If "local repair" is enabled and the destination is less than MAX_REPAIR_TTL hops away, RERR should not be sent to the

precursor nodes until after checking if a route to the destination can be found;

2. The destination sequence numbers from the RREPs of the intermediate nodes are smaller or equal to the the ones from the destination node. So the source would not accept the routes from RREPs that came from the intermediate nodes;
3. The route expiry time in OPNET is not updated when a node receives a data packet that came from a source that the node has no valid route entry. This can cause the route entry to the previous hop to end too early; and
4. The neighbour connection time in OPNET is not updated if the route of the received data packet's source does not exist in the routing table. This causes routes to become invalid even when it should be available.

It is perceived that the first problem identified above can be avoided if the "local repair" function is disabled. The second problem is addressed in Section sec:AODV - Destination-Only Flag Disabled Issue. The last two problems are addressed in Sections 4.4.4.3 and 4.4.4.4.

4.4.4.2 OPNET's AODV - Destination-Only Flag Disabled Issue

When the "destination-only" flag is disabled, the intermediate node is able to send a RREP back to the source if the node has a route to the destination. This may cause a problem where the destination can be using a different route to the intermediate node(s). In the scenario mentioned in Section 4.4.4.1, the destination ($[dest]$) node would be using the path, $[dest] \rightarrow [mob1] \rightarrow [static] \rightarrow [src]$, as it does not know about the path change, while the source and all the intermediate nodes ($[src]$, $[static]$-$[mob2]$) are using $[src] \rightarrow [static] \rightarrow [mob1] \rightarrow [mob2] \rightarrow [dest]$. The destination may not know about the new route. Although this does not seem to create much problems with this small network. This can be an issue when the network is big. If this causes problems to routing, the problem can be avoided by enabling the "destination-only" flag.

4.4.4.3 OPNET's AODV - Route Expiry Time Update Problem

In AODV, upon receiving a broadcast packet from a neighbour, the route expiry times for all the relevant routes using that neighbour as their next hop are updated. However, in OPNET's AODV, only the route expiry time of the neighbouring node gets updated. This is because the code to support route expiry time updates to all the relevant routes is missing. Thus, it causes the routes to drop out more frequently than it should, and hence more route setups were required. This can be fixed by adding a few lines of code to check all the routes in the routing table and update the route expiry time of the relevant route entries. The solution to this problem is given in Appendix A.3.

4.4.4.4 OPNET's AODV - Neighbour Update Problem

In standard AODV, when a HELLO packet is received, the route expiry time of all the routes that use the source of the HELLO packet for its next hop need to be updated. In OPNET, upon receiving a HELLO packet, only the route expiry time of the neighbouring node is updated, the route expiry time of all other routes using that neighbouring node are not updated. This causes link timeouts to occur, and eventually stop transmitting HELLO packets when all its neighbours' route expiry time have expired. The pseudo code in Appendix A.4.1 shows the modifications made to fix this problem. This problem has been rectified by OPNET, and a reference to the report of this problem and solution (Software Report ID 151780) can be found in OPNET's support center [114].

Furthermore, OPNET's AODV maintains a neighbour connectivity table that keeps track of all the neighbouring nodes' connectivity expiry times. A neighbouring node is kept connected by performing a neighbour connectivity update to update its expiry time when a data or a routing packet is received. If this time expires, the link will be considered to be disconnected, and hence the route to that neighbour will be removed from the routing table. Just like what happens when the route expiry time of the route entry to the neighbour ends. Therefore, whenever the route expiry time of the neighbour is updated, the connectivity expiry time in the neighbour connectivity table should also be updated. However in OPNET's model, when a data packet is received, the neighbour connectivity table is not updated while the route expiry time of the neighbour is, causing the neighbour connectivity expiry time to expire while the link is still valid according to the routing table, and often the expired neighbour is also physically available. This causes many RERR packets to be sent unnecessarily, and as a result, routes get disconnected, and hence, RREQ and RREP are sent to re-establish the routes, resulting a high routing traffic load on the network. This can be fixed by ensuring the connectivity expiry times in the neighbour connectivity table are also updated while the route expiry time of the route entry in the routing table is updated after receiving a data packet. Again, the solution to this problem is in Appendix A.4.2.

4.4.4.5 OPNET's AODV - Send RERR Problem

In AODV, when a link break is detected, the related nodes unicast/broadcast a RERR notifying their neighbours about their unreachable nodes due the link break. Each node that receives a RERR will check if there are any precursors that are sending to any of the unreachable nodes through the sender of that RERR packet. If there are precursors, the node would unicast/broadcast a RERR packet with its unreachable nodes and a list of unreachable nodes from the received RERR packet. However, with OPNET's AODV model, there are times when some precursors are not informed about the unreachable nodes. It is found that this problem occurs when a node is checking for precursors for a list of unreachable nodes, if the last unreachable node in the list only has one precursor, the node will send the RERR directly to that precursor regardless of how many other precursors for other unreachable nodes it has previously found. This problem was detected when a larger network was simulated with more traffic sending from different nodes. This problem can be fixed by ensuring that when there are one or more precursors, the RERR is broadcast. The pseudo code for the solution of this problem is in Appendix A.1.

4.4.4.6 OPNET's AODV - Precursor List Maintenance Problem

This problem in OPNET's AODV protocol causes a lot of packet drops in the network. This is because the precursor list is not maintained properly. In order to ensure the precursor list is maintained correctly, when a data packet for a destination arrives at an intermediate node, the previous hop of the data packet should be added to the precursor list of the destination's route entry if it is not already added. In OPNET, it is found that there are cases when a route to the destination was previously set up due to receiving a RREQ from the destination, and if there is a packet from the application layer arrives for that destination, no RREQ needs to be sent for route discovery, as it has already got a route to the destination. However, this creates a problem, because in OPNET the precursor list for the destination gets updated only when a RREP packet is received or sent, not when a data packet is received. So it is possible that the precursors have not been added to the precursor list of the destination in the intermediate nodes along the route. This problem can be prevented by adding some code that causes the node to update precursor list of the source route entry whenever a data packet needs to be forwarded to the destination is received. The solution to this problem, in pseudo code, can be found in Appendix A.2.

4.4.4.7 Modifications to AODV for Implementing AODV-PP

As explained in Section 4.3, there are differences between AODV and AODV-PP. Thus, a number of modifications have been made to the AODV model in OPNET to implement AODV-PP[17]. A summary of the modifications made to AODV include:

1. Adding a "Pathchange Time" field to the RREQ packet to enable the nodes to select the route with the longest lifetime;
2. Upon receiving a RREQ for a destination, the node broadcasts the RREQ if it has a longer estimated path-change time, and discard it otherwise. (The pseudo code for handling the arrival of the RREQ packet is illustrated in

[17]The pseudo code for the modifications listed can be found in Appendix B.

Appendix B.2);

3. After the first RREQ has arrived at the destination node, the node waits for a defined period of time for subsequent RREQs with a longer path-change time before sending a RREP;
4. When it is time for the destination to send a RREP, it selects the return hop that has the largest estimated path-change time to the source. (See appendix B.3 for the pseudo code to send a RREP packet upon an interrupt call at rrep_send_timer);
5. Upon receiving a RREP, store the route information from the RREP into the route entry to the destination, and change the existing route information of the route entry to the source with its pre-route information, then forward the RREP using the new route information. (For further details, see the pseudo code for processing the arrival of the RREP packet illustrated in Appendix B.4);
6. New routes are to be pre-fetched (i.e. broadcast a PreRREQ) before the estimated route expiry time is reached. (The pseudo code for broadcasting a PreRREQ can be found in Appendix B.5, and the code in Appendix B.7 sets the timer to broadcast the PreRREQ packet);
7. Additional route entry information are required to be stored temporarily during the "pre-route" stage to keep track of the latest best route for later use (when a RREP to the source is received);
8. If a RREP has not been received and the route entry is still valid, the pre-route data will be discarded; and
9. Whenever a data packet is received or being sent, the link lifetime and the route lifetime are updated. (The code for this is illustrated in the pseudo code in Appendix B.6).

The modifications listed above may increase the routing overhead traffic due to the extra RREQ packets that the intermediate nodes may need to be broadcast due to a higher path-change time is estimated. This may cause an increase of routing traffic during the RREQ-RREP phase. Routing traffic can be reduced by making the following modifications:

1. When an intermediate node receives a RREQ, it can wait for a short period

of time longer to see if more RREQs will arrive, then broadcast a RREQ with the longest path duration. However, this may further increase the end-to-end delay during path initialisation; and

2. An intermediate node may choose not to forward the subsequent RREQs that have a slightly longer path lifetime.

Though the two aforementioned methods for reducing routing traffic can be implemented in AODV-PP, they are not implemented. This is because not using these methods is closer to the existing AODV model, and it can ensure that the path with the longest lifetime to be found.

4.4.5 Application Layer Model

The application layer's process model that is chosen allows the user to specify the parameters for traffic generation. This enables the user to determine the destinations, the packet's inter-arrival times and the packet sizes of each node, which can either be fixed or randomly distributed.

This process model has been modified for data analysis purposes. It enables the destination to keep track of all the packets have been sent by the source and received by the destination, as well as the ones that are not received. This is achieved by allowing the source to inform the destination with the packet ID that is to be sent to that destination. When that packet is received by the destination, the packet ID information is removed, and the packet received counter is incremented.

4.4.6 Additional Statistics

Additional statistics are needed to record the data transmitted and received in AODV. Hence, code is added to the ip_rte_support external C code to trigger a statistics update in AODV while transmitting or receiving a data packet.

4.4.6.1 Statistics Collection

The existing statistics information collected in OPNET statistics is not enough for analysing the routing performance. Firstly, the route duration is not measured in OPNET's statistics. The route duration is measured by each source node to record the duration of the route, from the time when the route is first set up to the time when the route is either changed (a change in hop count or a change in destination address) or became invalid. However, if a node did not send any packet within the duration of the route, that node is not considered to be a source. In order to calculate the route duration in OPNET, a record of the following information is added to the AODV-PP model.

1. **Route Begins Time** - This is added to the routing table entry to keep track of the time when a route was set up for calculating the duration of a route; and
2. **Source Node Flag** - A flag to indicate whether the node is a source or not. This flag is reset to false if the route has been changed.

Moreover, the statistics collected in OPNET cannot be used to draw error bars. Therefore, codes have been added to the OPNET models for printing the statistics into a file, so that the statistics can be plotted in MATLAB. The following information is collected for detailed analysis:

1. Route duration;
2. Link changes;
3. Packet loss;
4. End-to-End delay; and
5. Routing traffic.

4.4.7 Testing the Models

Tests have been performed to ensure that MANET with AODV and Smart MAC models works the way it is expected. This is achieved by setting up a 5-node network scenario with three stationary nodes and two mobile nodes in a controlled situation, where the two mobile nodes have their own designated paths as shown in Figure 4.5 on page 79.

The tests have shown that there are some issues with this simplified node model. There is a long unpredictable delay between the time when a path can be obtained and when the route is actually found. This delay is caused because the source does not know when a path will become available, so the RREQs are often not sent as soon as a path can be obtained, instead, the source would wait until after its timer expires before sending a RREQ to search for the route. The delay varies because the timeout timer varies, and a path can become available at different stage of the RREQ process.

Through the tests, it is discovered that the route that was found was not always the shortest path when the OPNET AODV model was used. This is because in the AODV routing model, it has deliberately introduced some jitter while re-broadcasting the route request packet. This jitter is introduced to either provide some variation in packet processing time in different computers or to introduce a random delay so that the packets will be less likely to collide.

4.5 Simulation, Evaluation and Discussion

In order to conduct a comparative study between AODV and AODV-PP routing protocols to understand how predictive routing affect routing performance, a discussion on how AODV and AODV-PP are simulated and analysed is presented in this section.

4.5.1 Simulation Methodology

This section provides a description of the network scenarios used, the simulation parameters chosen for the scenarios, and the parameters configured for the node settings.

4.5.1.1 Network Scenarios

This chapter is aimed to explore the maximum benefits and the trade-offs between using AODV with perfect prediction and AODV routing protocols. The network scenarios used for the simulations consist of 24 nodes. The mobility model used for these nodes is RWP model. This model is chosen as it is more realistic compared to random walk motions that has no pause times. The parameters used for the RWP model is the same as that used in Section 3.3.1. In each network, 12 of the nodes move at one average speed, v_1; and the other 12 move at a different average speed, v_2. The nodes are placed randomly at approximately 2 km apart from each other to ensure that there is a route from one end of the network to the other end at all times. The first set of mobile nodes move randomly within a local area at average speeds of 0, 1, 2, 3 and $5m/s$. The second set of mobile nodes move randomly within a local area at average speeds of 1, 2, 3, 5, 10, 20, 30, 50, 60 and $80m/s$. 10 variations of scenarios are setup and executed for each pair of speeds v_1 and v_2. Each simulation runs for an hour. A sample of the network scenario is shown in Figure 4.9.

4.5.1.2 Node Configuration

The simulations are configured with the attribute settings for the physical layer, MAC layer, routing layer, the application layer, and tracking.

4.5.1.2.1 Physical Layer

In the physical layer, the transmission range for all the nodes is fixed at 4000m with no loss.

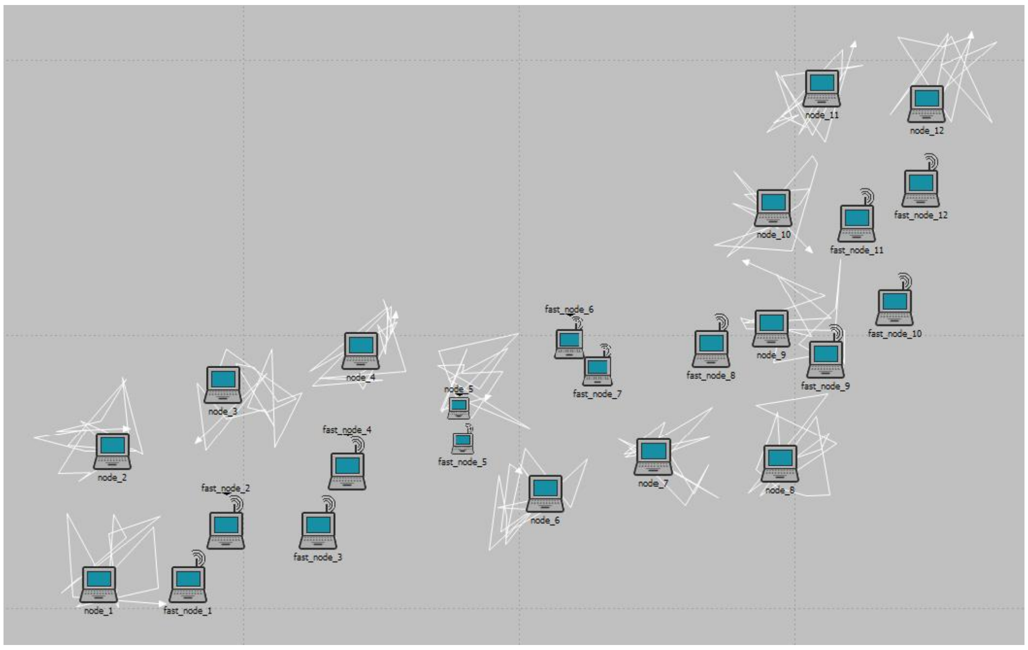


Figure 4.9: *OPNET - A sample of a 24-node network model scenario.*

4.5.1.2.2 MAC Layer

As mentioned previously in Section 4.4.3, “Smart MAC” is a ’near’ ideal MAC model that is supplied with OPNET. This model is used in place of a MAC protocol like WiFi to minimise the effects caused by the MAC layer, such as hidden node collision, has on the routing protocol’s performance. The default values are used for this model.

4.5.1.2.3 Routing Layer

AODV and AODV-PP are used in the simulations for comparison. Hence for both AODV and AODV-PP, the default AODV protocol settings set by OPNET are used, except for the following attributes:

1. **Destination-Only** - This is enabled, so that only the destination can send a RREP packet back to the source. This increases the chance of selecting

the best routes[18];

2. **Local repair** - This is disabled, so the intermediate nodes are not allowed to repair the route. Thus if a connection is lost, a RERR will be sent to the source to notify about the link break, so the source can send a RREQ to search for a new route;
3. **Hello interval** - This is set to 1 second;
4. **New AODV Routing** - This is for choosing to use either OPNET's AODV or the corrected AODV. For simulations with OPNET's AODV, set this to "disable"; and
5. **Simple Prediction Routing** - This is for choosing whether or not prediction is used. To use prediction, enable this attribute.

The "simple prediction routing" attribute is used to toggle between AODV and AODV-PP protocols. When the "simple prediction routing" attribute is disabled, AODV is used, and when this attribute is enabled, prediction is used.

4.5.1.2.4 Application Layer

Only 8 of the 24 nodes are generating traffic for a couple of randomly selected destinations. For each "traffic generation parameter", the "start time" for the traffic is $100s$, the "packet inter-arrival time" is exponentially distributed with a mean of $0.1s$, and the "packet size" is 1024 bits.

4.5.1.2.5 Tracking Configuration

In general, the default values are used. However, there is an option to disable the tracking model, called "track", if tracking is not required. Or if AODV-PP is used, enable the "track" attribute.

[18]The best route is determined by the route selection criteria. Therefore for AODV, the best route is the least number of hops, and for AODV-PP, the best route would be the one with the largest path-change time.

4.5.2 Comparison Metrics

The following metrics are used to compare the two protocols:

1. **Route Lifetime** - This is a measure of the duration or lifetime of a route in seconds without link changes;
2. **Number of Link Changes** - This is the average of the total number of link changes occur in each simulation. This measure is used to indicate whether the number of link changes in a network is proportional to the average speed of the nodes;
3. **Packet Delivery Ratio** - The number of bytes of application data packet received by the destination over the number of bytes of data packet sent by the source; and
4. **Route Setup Time** - The time required for route discovery, i.e. to establish and re-establish a route.

Note that routing overhead is not used for comparison. This is because the overhead traffic of AODV can be reduced substantially by utilising the MAC layer traffic to maintain the links of the active routes through cross-layer interactions, rather than using the HELLO packets. The author realised that a fair comparison of the overhead traffic between the routing protocols can only be obtained when this option is utilised. Hence, the routing overhead traffic is not analysed in this thesis. However, implementing the option of utilising the MAC layer traffic for route maintenance in OPNET's AODV model, and comparing the routing overhead traffic between the protocols, can be left for future work.

4.5.3 Results and Analysis

The results for comparisons between OPNET's AODV and the corrected AODV models, and between the corrected AODV and AODV-PP are analysed and discussed in Sections 4.5.3.1 and 4.5.3.2.

4.5.3.1 Comparisons between OPNET's AODV and the Corrected AODV

In ensuring that AODV-PP is actually comparing with a correctly functioning AODV, the aforementioned changes were made and verified by comparing OPNET's AODV model with the corrected AODV model. In this process, the corrected AODV protocol model is found to perform much better compared to OPNET's AODV in all the scenarios that were used. This can be observed from the average route lifetime graph and the average packet loss graph presented in Figures 4.10 and 4.11 respectively.

The graph in Figure 4.10 compares OPNET's AODV with corrected AODV with nodes moving at different average speeds. It shows that with OPNET's AODV, the average route lifetime for low speeds are not too different compared to the route lifetime for high speeds. However, with the corrected AODV, the average route lifetime can be seen to drop exponentially from much longer lifetimes for low speeds to short lifetimes for high speeds.

The problem in OPNET's AODV can also be verified by comparing the number of packet loss in both protocol models. It is shown in Figure 4.11, that the number of packet loss is so great for OPNET's AODV that the packet loss for the corrected AODV is insignificant in comparison.

The corrected AODV model is a fixed up model of OPNET's AODV model done according to the AODV RFC standard [16]. These two graphs (Figures 4.10 and 4.11) indicate that the low routing performance achieved by OPNET's AODV model is not suitable for making comparisons with AODV-PP, while the corrected AODV model will provide a fair comparison. Hence the results shown later in Section 4.5.3.2 are generated using the corrected AODV model rather than the original OPNET AODV model.

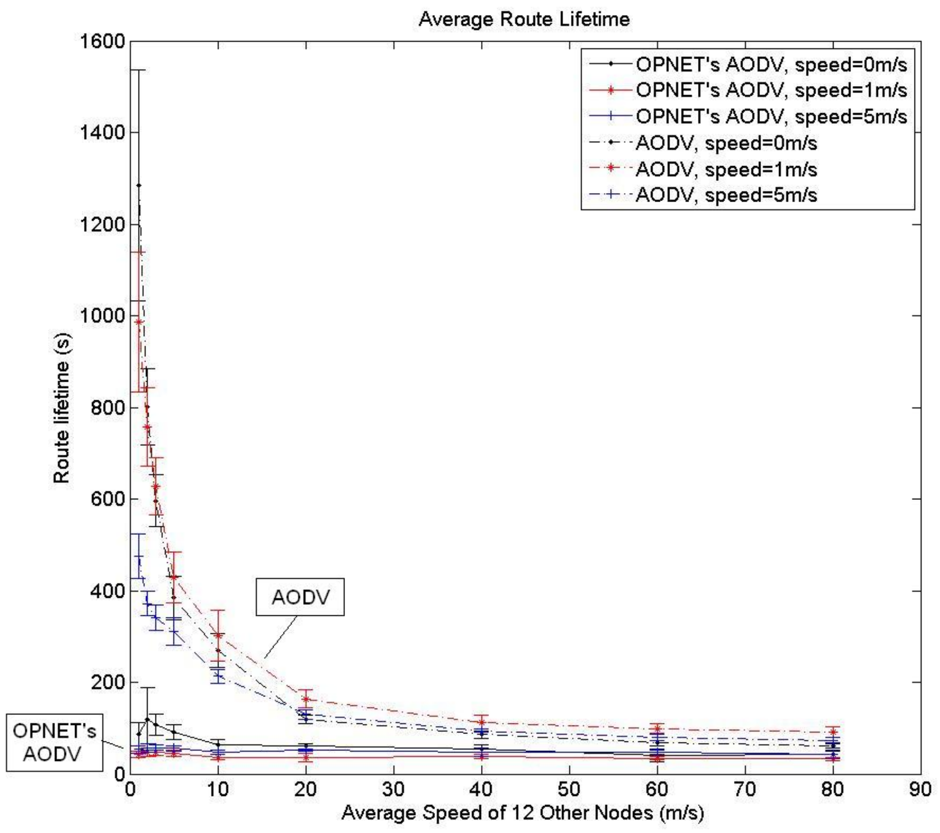


Figure 4.10: *Average Route Lifetime at varying speeds - OPNET's AODV vs Corrected AODV.*

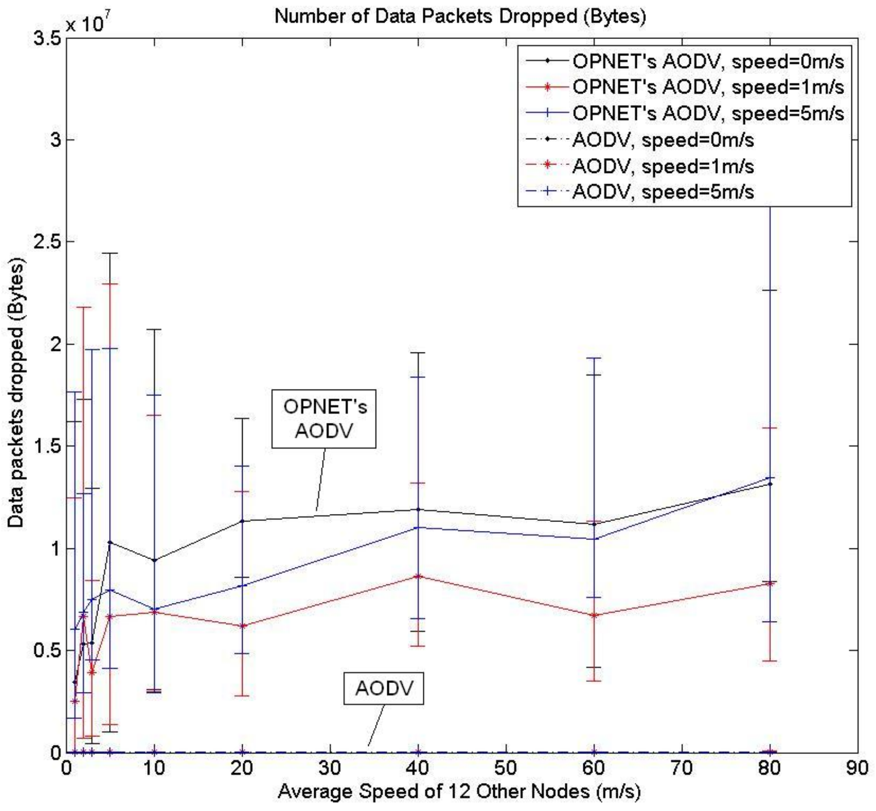


Figure 4.11: *Average Packet Loss in Bytes at varying speeds - OPNET's AODV vs Corrected AODV.*

4.5.3.2 Comparisons between AODV and AODV-PP

A comparison of the AODV and the AODV-PP protocols was made by comparing the average route lifetimes, the data delivery ratio and the route establishment times of the two protocols.

4.5.3.2.1 Average Route Lifetime

Figure 4.12 shows the average route lifetimes for AODV and AODV-PP when one set of nodes move at an average speed of 0, 1, 2, 3 and $5m/s$ against the average speeds of another set of nodes. The graph indicates that the average route lifetimes for AODV-PP is much longer than AODV for all cases. Although the route lifetime is greater with AODV-PP than with AODV, in the case where the shortest path (with the least number of hops) has the longest route lifetime, AODV-PP has no advantage over AODV.

Furthermore, Figure 4.12 indicates that as the speed of one set of nodes increases, the average route lifetime decreases for AODV, but it is not always the case for AODV-PP. This is because AODV selects the shortest path, so the network is more likely to have a greater number of link changes when the nodes have a higher average speed, as shown in Figure 4.13, and hence more route changes, i.e. shorter route lifetime. Whereas with AODV-PP, the path with the longest lifetime is selected, in which case, more link changes does not necessarily mean more route changes. Figure 4.13 also indicates that when both sets of nodes are moving, the number of link change increases as the nodes' speeds increase, however, if one set of nodes is stationary, there are more link changes compared to cases where both sets of nodes are moving.

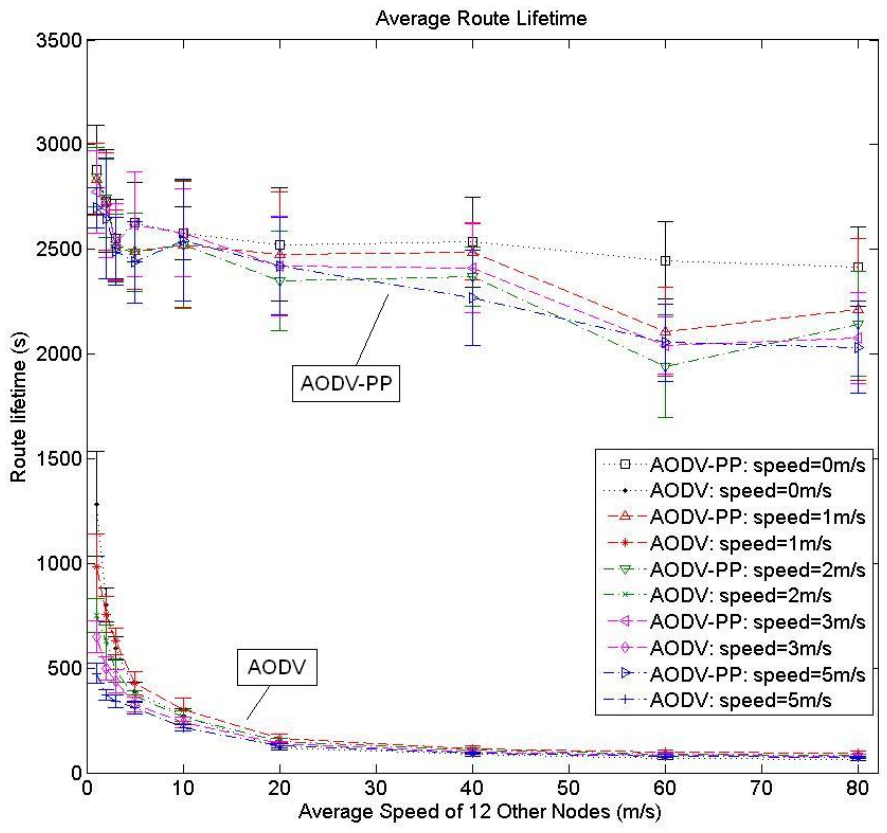


Figure 4.12: *Average Route Lifetime at varying speeds - AODV vs AODV-PP.*

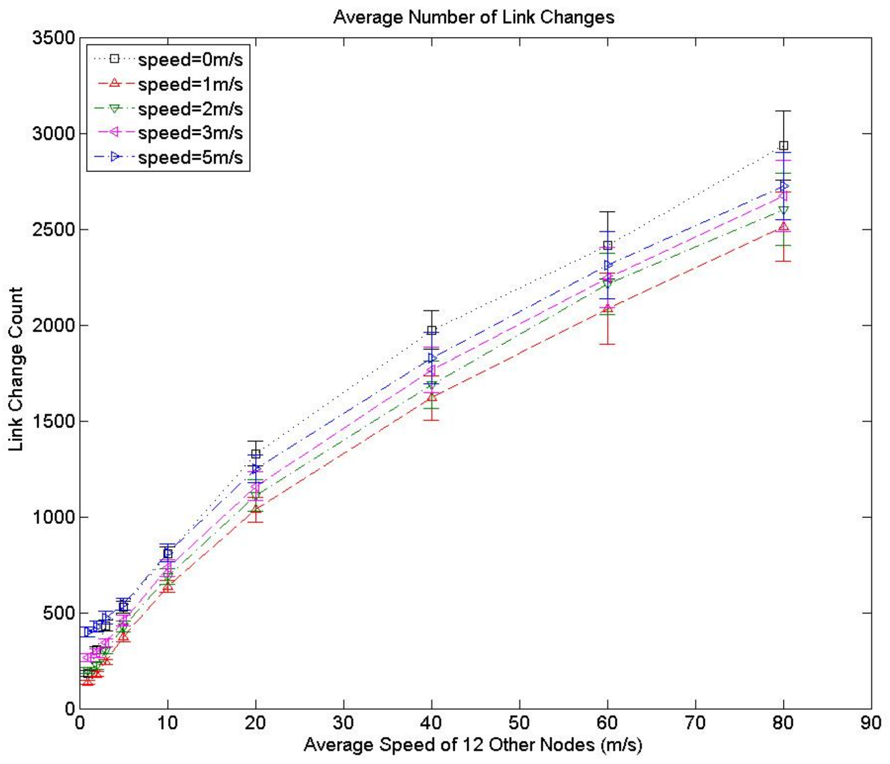


Figure 4.13: *Average Number of Link Changes at varying speeds.*

4.5.3.2.2 Packet Delivery Ratio

Figure 4.14 shows the average data packet delivery ratio for the two routing protocols when one set of nodes move at an average speed of 0, 1, 2, 3 and $5m/s$ against the average speeds of another set of nodes. The graph shows the number of bits received by the destination over the number of bits sent by the source, so it scales between 0 (meaning no transmitted packets has been received) and 1 (meaning 100% of the data transmitted has been received). The graph shows that when perfect prediction is used the packet delivery ratio remains at almost 100% delivery regardless of the speeds of the nodes (the lines on the graph are overlapped). Whereas without prediction, the packet delivery ratio is slightly lower in percentage. This is because when there is no prediction, the rapid link changes in high mobility networks causes more route changes, which increases the packet loss. But when prediction is used, the routes are re-established before the route breaks, hence no packet loss. Therefore, the more mobile the network is, the better the AODV-PP is compared to AODV.

Another point to notice from Figure 4.14 is that the difference in packet loss between AODV and AODV-PP is not that great. The reason for this is that for AODV, the time between the link being disconnected and when the source node is notified about the broken route (which can be estimated to be around 2 seconds according to the AODV protocol) is relatively small compared to the lifetime of the route, and the traffic load is low, hence there is not much packet loss. This is why the data delivery ratio remains quite high on the graph even for the AODV scenarios with some fast moving nodes. Nevertheless, 2 seconds without connectivity (which causes packet loss) can cause congestion problems in Transmission Control Protocol (TCP)[19], because TCP assumes that packet losses are caused by congestions [115, 117].

[19]TCP congestion problem is a well known problem, and has attracted many researchers, and thus many solutions have been proposed [115]. However, most proposed solutions still require further development. Nevertheless, TCP Vegas [116] can be a feasible solution to this problem as it performs effectively in most cases. It utilises the round trip time of the packets rather than using packet loss to detect congestion.

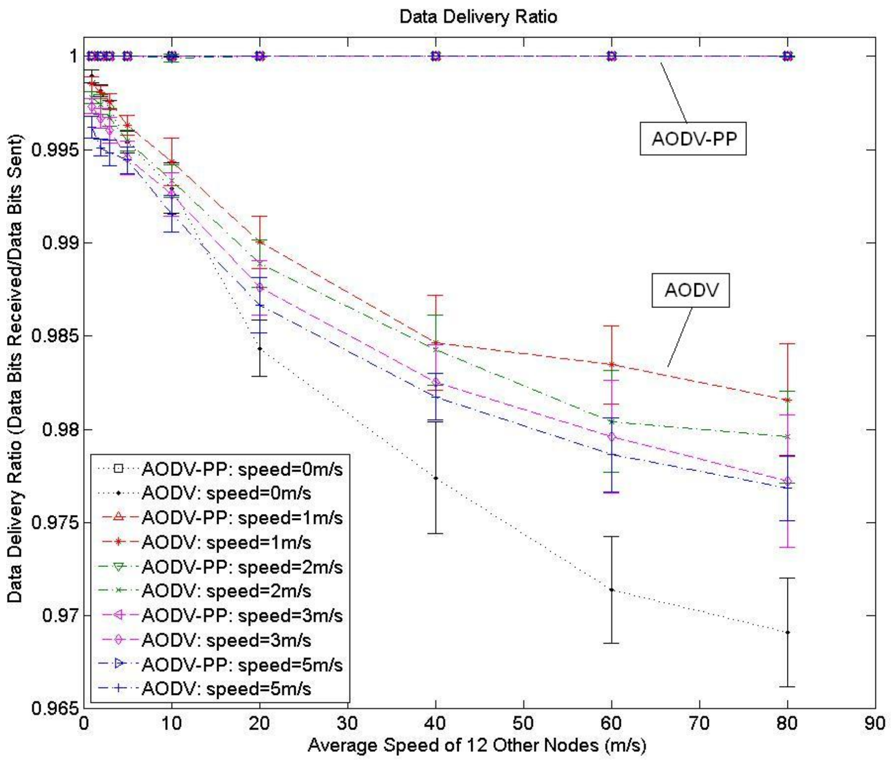


Figure 4.14: *Average Data Packet Delivery Ratio at varying speeds - AODV vs AODV-PP.*

4.5.3.2.3 Route Setup Time

The average route setup time shown in Figure 4.15 indicates that AODV-PP's average route setup time is longer than AODV. This is due to the long waiting time at the destination node of AODV-PP upon receiving a RREQ in an attempt to wait for subsequent RREQs with longer route lifetimes. However, because a route can last longer in AODV-PP, it does not need to re-establish a route as often, thus the total route setup time of AODV-PP is less than AODV, as presented in Figure 4.16.

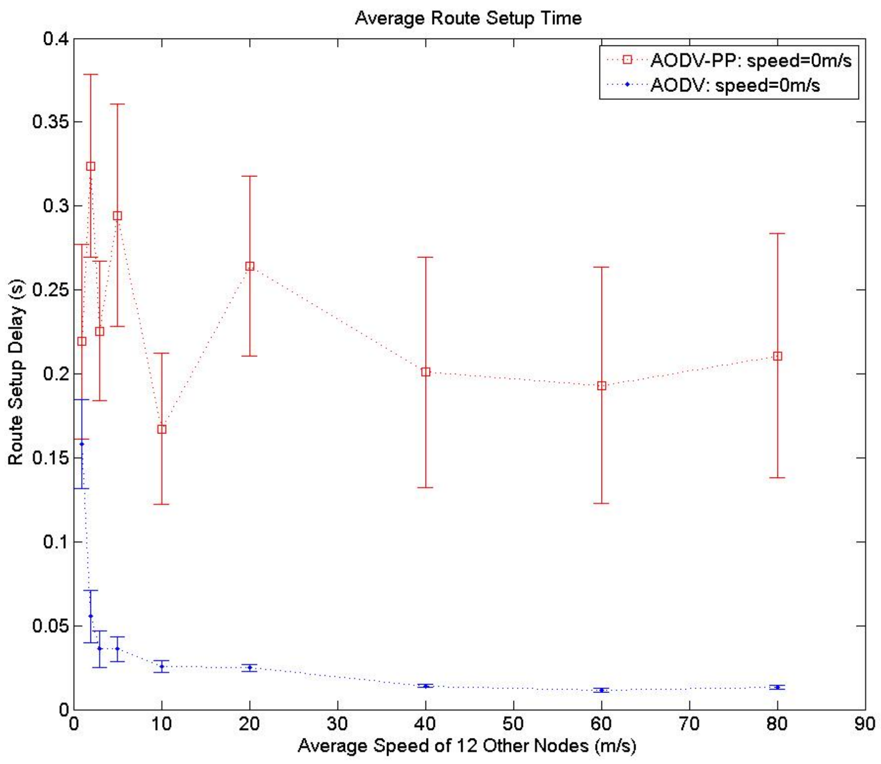


Figure 4.15: *Average Route Setup Time at varying speeds - AODV vs AODV-PP.*

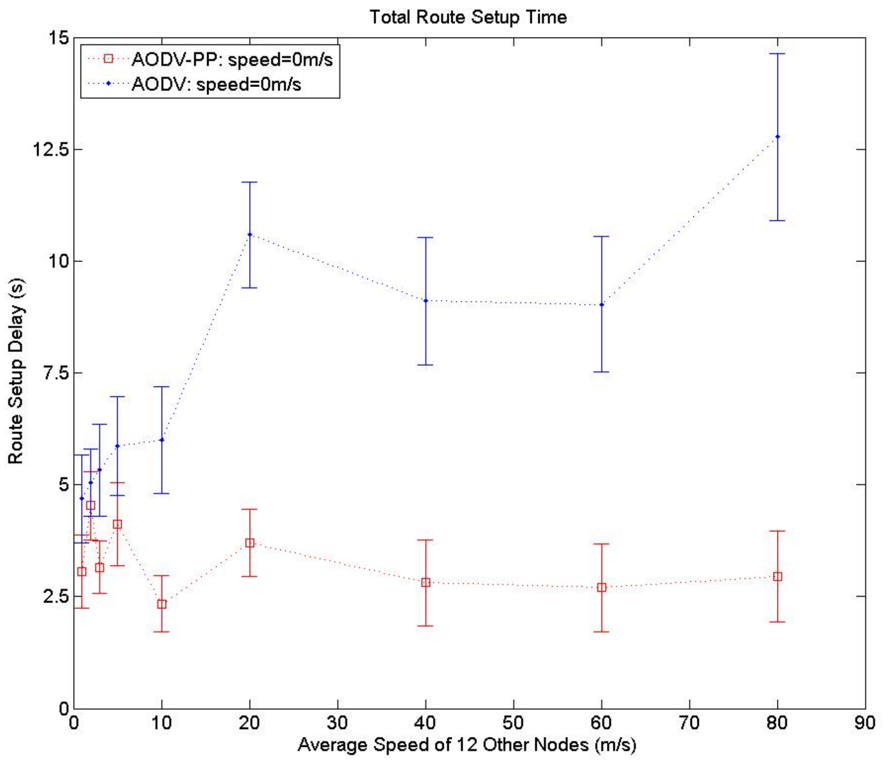


Figure 4.16: *Total Route Setup Time - AODV vs AODV-PP.*

Overall, these results show that when prediction is used, the duration of the routes are substantially longer and fewer routes change compared to without prediction. This indicates that the routes of the network are more stable when prediction is used. Furthermore, the packet delivery ratio and throughput is greater with prediction due to its ability to find a new route before link breaks. However, the trade-off for using prediction is that it increases the end-to-end delay as a more long-lived route may not be the shortest path. Moreover, the route set up time may be higher too, because the protocol has to wait for more RREQs before deciding which path to choose. Though it is possible to send a RREP as soon as a RREQ is received, more RREP will need to be sent if any subsequent RREQ received came from a more stable path. In which case, there may be more route changes during route setup.

4.6 Conclusion

The results in Chapter 4 has shown that AODV can be modified to utilise link duration prediction information to enable the destination to select a route that has the longest link duration (or the most stable). According to this study, using perfect prediction can maximise the route duration, minimise the number of route changes and increase throughput. However, there is a trade-off of increasing route set up time and end-to-end delay. Overall, this study indicates that routing with link duration prediction can be more suitable for networks or services that requires a more stable route, and that it would be useful when the nodes in the network are moving at different speeds independently causing some links to be less stable than other.

Although this study has provided some insights into how well routing can be performed with accurate link duration prediction, it does not indicate whether it will be useful for real life situation where prediction accuracy can vary. Hence, there are still a number of questions that need to be investigated. These include:

1. What happens to the routing performance when there are errors in the link

duration predictions?

2. How well can predictive routing tolerate errors in the prediction?
3. What level of prediction accuracy is required to improve routing performance?
4. Is it feasible to use prediction for routing, and if so, what are the benefits?

The next chapter of this thesis is intended to provide more in depth understanding to the study of using prediction in routing, which answers the above questions.

Chapter 5

AODV with Prediction Update

5.1 Introduction

A performance comparison has been made between AODV and AODV-PP in Chapter 4. The work in Chapter 4 shows that by adding accurate link duration prediction to AODV will generally lead to improved routing performance that, in many cases, find a much more stable route. However, in reality, predictions are not always accurate, as indicated in Chapter 3. In what circumstances would prediction still be useful when there are errors in the predicted paths? The purpose of this chapter is to study how prediction errors affect the routing performance. This has been achieved by evaluating the performance of AODV with prediction when errors are present, and how well can predictive routing tolerate errors compared to AODV and AODV-PP.

For the purpose of this study, a further development to the AODV-PP routing protocol has been made. The initial change was to create a new variation of the AODV routing protocol that allows the nodes to select a route with the longest predicted route duration. Now new functionality is added, which enables update link duration changes to take place where necessary. This novel routing protocol is called, AODV with Prediction Updates (AODV-PU), and it will be explained in more detail in Section 5.2. Section 5.3 describes the modelling of the AODV-PU protocol and the tracking model. The simulation method, and the evaluation and discussion of the results are explained in Section 5.4.

5.2 AODV with Prediction Update

In a real MANET, nodes move around randomly and signal strength at platforms' receivers varies due to changing signal attenuation caused by different transmission paths and changing environments. Accurately predicting these changes is a very difficult exercise. Thus, it is highly likely that there will be errors in the link duration predictions made. In Chapter 4, it has been shown that stable routes with little link changes can be selected if the prediction is very accurate, and that knowing the link change time allows a new route to be found before a link breaks so there will be no packet loss. However, in the presence of errors in the link duration predictions, it is uncertain how the stability of the routes and hence the packet loss will be affected.

In the previous chapter, an enhanced version of AODV, AODV-PP, enables the predicted route duration value to be broadcast through the network via RREQ and RREP. This protocol only works based on the assumption that the link duration predictions are accurate. This is because with perfect link duration prediction, there is no route expiry time update required, as there is no error in the initial prediction. It would fail in the presence of errors, as it cannot inform the source if the predicted link duration has changed. In order to accommodate this change, the protocol is required to be able to forward the link and route duration updates back to the source. A new variation of AODV-PP routing protocol, AODV with Prediction Update (AODV-PU), has been developed to allow route duration prediction updates for any destination to be propagated back to the source before the link breaks. This ensures a more graceful route updating process than having the route break unexpectedly.

New functionality has been added to AODV-PP to create AODV-PU in order for the source to be informed about any prediction updates. This involves letting the intermediate nodes process and send new route duration updates back to all the sources for all the destinations that are caused by a change in link duration, and allowing the source to process any changes in route duration and estimate the time to search for a new route accordingly. The other existing functions of AODV-PP remains unchanged.

Prediction updates can be accomplished using the following steps. The first step is to continuously monitor and estimate the link duration of each hop, as explained in more detail in Section 5.2.1. Second is to determine when to send and inform the source(s) to update the link or route expiry time if the change in link duration affects the route duration, as described in Section 5.2.2. The final step is to send and propagate a route expiry time update (RUPDATE) packet back to the source(s) about the new route expiry for a specific destination is explained in Section 5.2.3.

5.2.1 Link Duration Estimation

In AODV-PP, the link duration is calculated according to the pre-determined mobility model used for perfect accuracy. Now this study looks into how prediction errors affect routing performance, therefore errors need to be introduced into the prediction models. There are two simple ways to achieve this. One way is to add some errors to the perfectly calculated link durations. Alternatively, use an existing link duration prediction method that would provide a range of accuracy when the nodes move and change directions at different rates.

Although any link duration prediction method can be utilised by the AODV-PU protocol, for simplicity, the link duration in this study is estimated using the Link Expiration Time (LET) formula proposed in [2–4]. This formula assumes that the mobile nodes are moving at a constant velocity. If the transmission range is r, the locations of node i and node j are (x_i, y_i) and (x_j, y_j), the speeds of node i

and node j are v_i and v_j, and the directions of node i and node j are θ_i and θ_j respectively, the LET between node i and node j, $T_{i,j}$, can be computed using Equation 5.1.

$$T_{i,j} = \frac{-(ab+cd) + \sqrt{(a^2+c^2)r^2 - (ad-bc)^2}}{(a^2+c^2)} \tag{5.1}$$

where a, b, c and d are:

$$\begin{aligned} a &= v_i \cos\theta_i - v_j \cos\theta j \\ b &= x_i - x_j \\ c &= v_i \sin\theta_i - v_j \sin\theta j \\ d &= y_i - y_j \end{aligned} \tag{5.2}$$

Upon receiving a packet, the packet arrival time and the location information of the sender and the receiver are retrieved from the simulation model. When a packet from the same sender is received, and the packet arrival time and the location information of the sender and the receiver are retrieved, the link duration is then calculated using LET. For simplicity of implementation, this study assumes the link duration estimation is performed upon receiving a data packet. This assumption is realistic for the simulations presented in this chapter, as the data packets are transmitted regularly from the source to the destination without very big time gaps.

However, such an assumption may not be feasible for real networks, as the traffic between any two nodes may not be regular. In this case, the link duration can be estimated upon the arrival of other types of packets such as the management and control frames of the MAC layer. Note that this is possible since the link duration prediction can be estimated upon the arrival of any type of packets from any layer using cross-layer methods.

It is also possible that this link duration estimation technique may not be suitable for a real network, because it may require extra bandwidth to allow the location information to be broadcast to the neighbouring nodes, which is not accounted for in the simulations of this study. Though, the location information can be added to the routing packets, such as the HELLO packets of the AODV protocol, or the management frames, like the BEACON frames of the MAC protocol[1], more bandwidth will be required for broadcasting. Alternatively, the link duration estimation can be achieved by measuring the RSS and the SNR of the incoming packets to estimate the location of the previous node or the lifetime of the associated link.

5.2.2 Set Route Expiry Time Update Timer

In the case with perfect prediction, the source knows exactly when the route will end, so it can always send a PreRREQ a short time before the route becomes unavailable. In circumstances where prediction is not accurate, a link may break before the predicted route expiry time, and the source would not know about the link break until seconds later when a RERR is received. Or if the route is later found to be longer than the original prediction, the source would send a PreRREQ even if the route remains available for a longer duration. Therefore, there is a need to improve the AODV-PP protocol so that the source can be informed if there is a change in the predicted expiry time at any link along a route. Hence AODV-PU is proposed. In this protocol, if the estimated link's lifetime has changed, the two associated nodes will broadcast a route expiry time update (RUPDATE) packet to inform their precursors about a change in route expiry time.

A RUPDATE packet consists of two fields, one contains the new route duration to the destination, and the other is the associated destination address. The structure of this packet can be found in Appendix C.1.1.

If the route duration to a destination has changed, a RUPDATE packet will be sent to inform its precursor nodes. A RUPDATE packet can be sent before the

[1]BEACON frames are the management frames of WiFi that are broadcast periodically

route expires, so that there will be enough time for the source to decide when to find a new route to the destination.

There are two different times one can set to send a RUPDATE packet:

1. One option is to send an update whenever a route expiry time has changed. However with this method, if the estimated link duration changes regularly, there would be a lot of updates flooding the network.
2. The other option is to send the RUPDATE packet before a link breaks or before the duration of the link gets extended. This can be achieved by setting a timer to send a RUPDATE packet at either:
   (a) Before the end of the latest estimated route expiry time (new_route_expiry_time[2]) if the new_route_expiry_time is shorter than the original route expiry time, so that the RUPDATE packet can be sent before the link breaks; or
   (b) Before the original route expiry time ends if the original route expiry time is shorter than the new_route_expiry_time, so that the RUPDATE packet can be sent to extend the route expiry time to the new_route_expiry_time, for this and the precursor nodes.

Since the first method listed above is not very efficient with large overheads, the second method is chosen for AODV-PU. With the second method, it is important to determine the appropriate time to send the route expiry time update to the senders, as it can affect the performance of the routing protocol. In order to ensure that the RUPDATE packet can reach the source node with enough time to re-establish a route (i.e. send a PreRREQ and receive a subsequent RREP) before the route expires, a RUPDATE packet should be sent at least 1.5 times of the RING_TRAVERSAL_TIME[3] before the end of either the existing or the new route expiry time. Thus, to ensure that everything is covered, extra time is given to allow some inaccuracies in the prediction.

[2]new_route_expiry_time is one of the new variables added to the route entry to implement AODV-PU. See Table D.4 in Appendix D.2.

[3]RING_TRAVERSAL_TIME is used to set the timeout for receiving a RREP packet, and is defined in Table D.1 of Appendix D.1

Hence, the route_expiry_update_timer[4] ($t_{RexpUpdate}$) is defined to be 2 $\times$ MAX_RING_TRAVERSAL_TIME ($t_{MaxRingTrav}$) less than the minimum value of the route_expiry_time (t_{Rexp}) and the new_route_expiry_time (t_{NRexp}), as shown in the following equation:

$$t_{RexpUpdate} = min(t_{Rexp}, t_{NRexp}) - 2 \times t_{MaxRingTrav} \tag{5.3}$$

where $t_{MaxRingTrav}$ (MAX_RING_TRAVERSAL_TIME) is defined in equation 4.4.

Although a coefficient of 2 is selected for this study, this value can be dynamic. It can be varied according to the accuracy of the prediction. If the prediction is very accurate, smaller value can be used. Otherwise, larger value may be required.

5.2.3 Route Expiry Time Update

A RUPDATE packet is broadcast to perform route expiry time update, which is used to inform the source about the change in route expiry time for a specific destination. This packet consists of the destination address and the route expiry time from the sender (intermediate node) to the destination, as illustrated in Appendix C.1.1.

When it is time for the intermediate node to send RUPDATE, the node checks if there are any precursor nodes in the route entry of the destination. If there are no precursor nodes, no RUPDATE needs to be sent, otherwise, a RUPDATE packet is broadcast.

Upon the arrival of a RUPDATE packet at a neighbouring node, it checks if it has a valid route entry for the destination that uses the sender of the RUPDATE packet as its next hop to the destination. If so, the node updates the new route

[4]route_expiry_update_timer is the time to send a RUPDATE packet, and is one of the new variables added to the route entry to implement AODV-PU. See Table D.4 in Appendix D.2.

expiry time of the route entry. Otherwise, it ignores the packet. If there is one or more precursor nodes in the route entry, set the route expiry time update timer according to the way described previously to re-broadcast the RUPDATE packet. Furthermore, if the receiver is the source for the destination specified in the RUPDATE packet, it sets a timer to perform PreRREQ as explained in Section 4.3.2 of Chapter 4.

5.3 Modelling and Implementations

As with Chapter 4, OPNET Modeler has been used to implement models and execute simulations for this study. The models that have been modified to accomplish this study include the tracking process model and the AODV process model.

5.3.1 Modifications of the Tracking Model

In the previous chapter, the tracking model was implemented with the prime purpose of calculating the perfect link duration of a communication link with any other node in the network. Section 5.2.1 has shown how errors are needed for this study. Thus, an option is added to allow users to choose LET for the prediction method, and an external function is added to the tracking model to calculate the LET.

Unlike perfect predictions, true predictions can only be made when there is data input. Hence, a constant update of the nodes' locations need to be observed and recorded. This can be achieved via storing the last one or two locations of all the nodes in the tracking model. This information allows the link duration to be calculated using LET when needed.

5.3.2 Modifications for AODV-PU

The modifications made to AODV-PP in OPNET comprises of the following:

1. Upon receiving a data packet, if there is a valid route to the destination, predict the link expiry time of the next hop and the route expiry time of the destination. Then if the link expiry time is different to the one in next hop's route entry, set the route_expiry_update_timer and the new_route_expiry_time in the route entries of the next hop and all other destination nodes that use this hop as their next hop of their route, and for each of these route entries, set an interrupt to send a RUPDATE packet at the time specified in the route_expiry_update_timer. Otherwise, if the route expiry time is different to the route entry of the destination, set the route_expiry_update_timer and the new_route_expiry_time in its route entry, and set an interrupt to send a RUPDATE packet at the time specified in the route_expiry_update_timer. (See Appendix C.2 for the pseudo code for setting a timer to update route expiry time for the appropriate route entry upon receiving a data packet);
2. Added some code (in lines 19 to 24 in Code B.2.1) to the predict_linkchange_time() function in Appendix C.2 to allow LET to be used for link duration prediction method instead of perfect prediction;
3. Added the following parameters to the route entry structure:
    (a) **new_route_expiry_time** - used to record the new route expiry time, so that the current route expiry time will be updated to the new route expiry time when RUPDATE is sent;
    (b) **route_expiry_update_timer** - used to keep a record of the time when a RUPDATE packet needs to be sent; and
    (c) **last_route_expiry_update_timer** - the time when a RUPDATE packet is sent is recorded so that a RUPDATE packet will not be sent if one has previously been sent within a short period of time. This is to avoid too many RUPDATEs being sent unnecessarily if the predicted path-change time changes frequently.

   A list of all other variables added to the route entry to accomplish route expiry time updates in OPNET can be found in Table D.4 of Appendix

D.2;

4. Add a new RUPDATE packet structure, which consists of the new route expiry time and the destination address, as shown in Appendix C.1.1;
5. Add a function to send a RUPDATE packet to the precursor nodes when the route_expiry_update_timer is up. (The pseudo code for this can be found in Appendix C.3); and
6. Add a function to process the RUPDATE packet when it arrives. (The pseudo code for processing the received RUPDATE packet can be found in Appendix C.4).

5.4 Simulation, Evaluation and Discussion

This section consists of three parts. The first one provide information on the simulation scenarios used and the simulation configurations, the second part explains the metrics used, and the last part discusses and analyses the results of the simulations.

5.4.1 Simulation Methodology

The aim of this chapter is to show what happens to the routing performance in the presence of link duration prediction errors. Thus, this can be achieved by using the scenarios with 12 stationary nodes and 12 mobile nodes with different speeds, and RWP model as their mobility model, as mentioned in Section 4.5.1.1 of Chapter 4. The reason for choosing only the scenarios with 12 stationary nodes is so that some predictions can be more accurate than others, which is similar to a more realistic situation where some nodes may be more predictable than others.

The node's attribute settings for the simulations in this chapter are the same as that discussed in Section 4.5.1.2 of Chapter 4. However, for AODV-PU simulations, the "prediction method" attribute in "tracking" is configured to use "LET".

5.4.2 Comparison Metrics

The following performance metrics are used to compare AODV and AODV-PU:

1. **Route Lifetime** - This is a measure of the duration or lifetime of a route in seconds without link changes; and
2. **Packet Delivery Ratio** - The number of bytes of data packet received by the destination over the number of bytes of data packet sent by the source.

5.4.3 Results and Analysis

This section provides the results from the simulations performed according to the method and the evaluation metrics defined in Sections 5.4.1 and 5.4.2 respectively. Note that the accuracy of the prediction is not controlled in the simulations of this chapter. However, the prediction accuracy is determined by the average speed of the nodes. Although the accuracy is not measured, it can be achieved by using the methods described in Chapter 3 if required.

The graph in Figure 5.1 shows the average route lifetimes for AODV, AODV-PP and AODV-PU when a set of 12 nodes are stationary while the other set of 12 nodes are moving at different speeds. The graph indicates that the average route lifetimes for AODV-PU is better than AODV when some nodes are moving at low average speeds. However, at high average node's speeds of $10m/s$ or more, AODV-PU's performance is similar to that of standard AODV. The reason for this is that at slow average speeds, there are more accurate link duration predictions, and the nodes are able to choose a more stable route with a longer duration during route setup. Whereas at higher speeds, there are more chances for the link duration predictions to be wrong, hence the chosen route is less likely to be the most stable. Although the average route lifetimes for AODV-PU is better than

AODV, it is far worse than AODV-PP. This is due to the reduction in accuracies of the route lifetime predictions when LET prediction is used. This is indicating that the more accurate the initial prediction is, the better it performs.

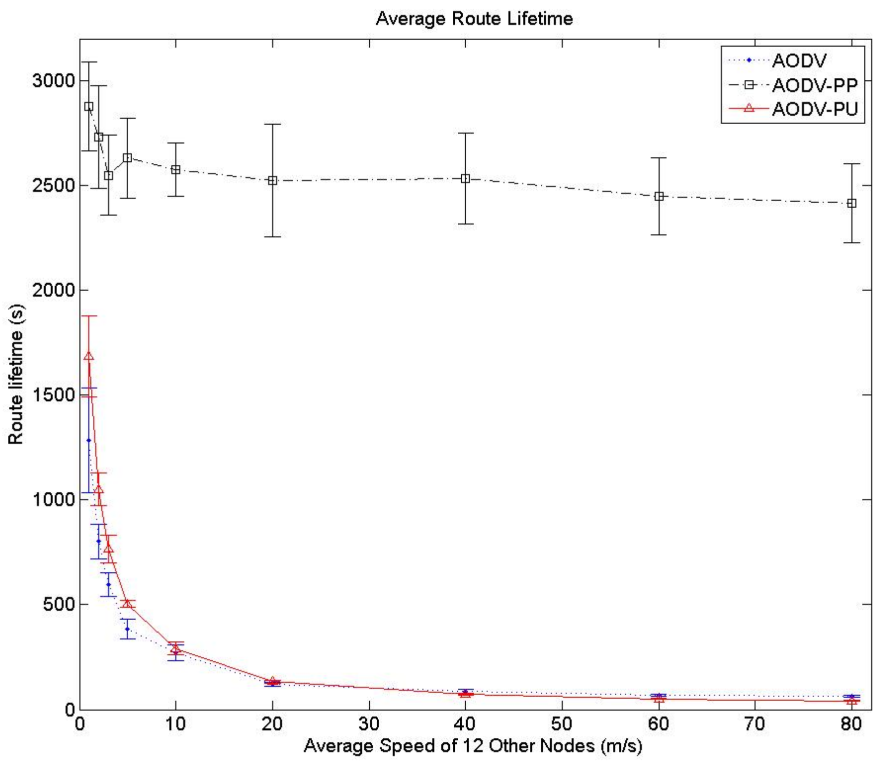


Figure 5.1: *Average Route Lifetime for AODV, AODV-PP and AODV-PU.*

The results presented in Figure 5.1 also indicate that at high speeds the average route lifetime of AODV-PU is just as short as or may possibly be shorter than AODV. This is because when the route duration is predicted to be longer than the actual duration, if one of the links is later predicted to be shorter, a route update will be sent to notify the source to send a PreREQ at PRE_ROUTE_START_TIME. Provided that the accuracy of the prediction at route setup is far from accurate, it is very unlikely that it will find any more stable route. Furthermore, with highly mobile nodes, it is possible that some links may only be disconnected for a short period of time. Therefore, with AODV, this time may not be long enough for the neighbouring nodes to notice the fault of the link to inform the source for route re-establishment. Whereas with AODV-PU, a disconnection could be predicted a short time before it breaks, hence an update would be sent to inform the source, but the re-connection of the link cannot re-establish the old route. Hence it is possible that the average route lifetime for AODV can be longer than AODV-PU.

The average data packet delivery ratio for the three routing protocols is shown in Figure 5.2. This graph indicates that when prediction is used the packet delivery ratio is very close to 1, which means it is almost perfect with no packet loss, even though there were errors in the prediction. Whereas data packet delivery ratio of AODV drops to an average of around 0.969 at high average speeds of $80m/s$. The near perfect result for AODV-PU is similar to the packet delivery ratio results of AODV-PP, which indicates that packet loss can be substantially reduced with prediction updates regardless of the high number of route changes due to short route lifetimes. The good data packet delivery ratio in AODV-PU is due to regular link duration predictions occurring at the intermediate nodes, and route updates were sent to inform the source of any link duration changes, enabling routes to be re-established before routes become unavailable. Unlike AODV, AODV-PU is able to reduce the packet loss caused by unnoticed link drop outs. Nevertheless, this cannot eliminate all packet losses, as predictions can sometimes be inaccurate, and hence it may be possible that the route updates cannot reach the source in time to search for a new route.

These results indicate that when the prediction is highly inaccurate, transmitting the predicted route lifetime with the RREQ/PreRREQ packets during route setup is not entirely useful in selecting a more stable route. Prediction seems to be more useful when the prediction is reasonably accurate. However, in the presence of highly inaccurate predictions, link duration prediction can still be useful between neighbouring nodes to maintain the availability of the route, and allow the source to establish a new route before it becomes unavailable, as this can prevent excessive packet loss.

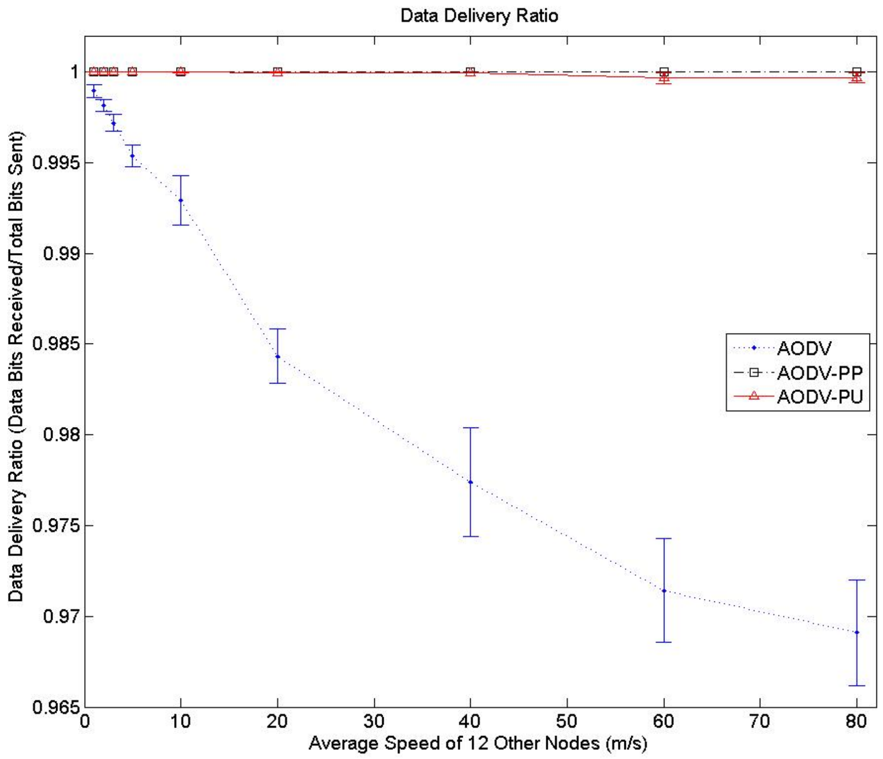


Figure 5.2: *Average Data Packet Delivery Ratio for AODV, AODV-PP and AODV-PU.*

5.5 Conclusion

This chapter has provided an insight into how to incorporate prediction information into AODV to perform route selection and route updates. It has been shown that by using link duration prediction in AODV-PU, the routing protocol is able to find a new route before the previous route becomes obsolete. Its ability to re-establish a route before it breaks can reduce packet loss even if the prediction is highly inaccurate. However, it is found that in the presence of large link duration prediction errors, it fails to find a stable route, as finding a stable route requires more accurate link duration prediction during the route establishment or re-establishment stages.

This research shows that using prediction for routing may not always be suitable. This routing protocol is shown to be suitable for networks that are sensitive to packet loss. In cases where link durations are more accurate, AODV-PU is also able to find a stable route. However, the trade-off for this protocol is similar to that of AODV-PP, which requires a longer route establishment time, and the selected route may have a longer end-to-end delay depending on the network topology and the mobility of the nodes.

5.5.1 Further Improvements for AODV-PU

Although AODV-PU has reduced the packet loss issue in AODV, it is not perfect, as predictions may not be accurate. Nevertheless, there are ways in which AODV-PU can be modified and enhanced, which may possibly improve the routing performance.

AODV-PU can be modified and enhanced in the following ways:

1. Using local route repair when there is an early link break rather than sending a route update back to the source. However, the trade-off is that this option may reduce the chance of finding the most stable route, as a

local route repair is accomplished from an intermediate node; and

2. Enabling intermediate nodes to respond to a RREQ/PreRREQ with a RREP if it has a route to the destination to reduce the RREQ/PreRREQ traffic. However, this may reduce the chance of finding the most stable route. This option may be better for scenarios with constant route changes and less accurate predictions.

Chapter 6

Conclusion

The benefits and the trade-offs of using prediction in AODV routing protocol have been identified in this thesis. The results of this thesis have shown that with perfect prediction, a significant improvement can be made in terms of routes' lifetime with almost no packet loss. This thesis has also identified the problem with predictive routing in that it can be difficult to select a long-lived route when prediction is not perfect. This is due to the difficulties in making near accurate predictions far ahead of time. However, prediction can still be effective for route maintenance, as route maintenance only requires short-term predictions, which are more accurate compared to long-term predictions. In accomplishing this, two new routing protocols, AODV-PP and AODV-PU, have been proposed.

Although the studies in this thesis have shown that prediction can enhance routing performance with accurate prediction, it can be seen from Chapter 3 that it is difficult to always make accurate predictions, especially when predicting far into the future. Therefore, in cases where prediction is not so accurate, it can shorten the lifetime of the routes considerably. Chapters 4 and 5 of this thesis have analysed the difference in routing performance between perfect and inaccurate prediction. In achieving this, two new variations of the AODV routing protocol, AODV-PP and AODV-PU, have been implemented.

In Chapter 4, it has been shown that AODV can be modified so that link duration predictions can be used for selecting the most stable route when predictions is accurate. The creation of the new AODV-PP protocol has been simulated in the OPNET environment, and has lead to realistic results. The results from this chapter indicate that with accurate predictions, stable routes can be found with a trade-off of longer delays in route discoveries.

The proposed AODV-PU protocol in Chapter 5 has demonstrated that additional functions can be added to AODV-PP to allow route duration updates to be sent from the intermediate nodes back to the source node to inform the source of the route duration changes. This enables new routes to be established before the old ones become unavailable to minimise packet loss. Nevertheless, although AODV-PU shows a little bit of improvements in average route lifetime over AODV with slow mobility, the poor average route lifetime results for AODV-PU compared to AODV-PP indicate that the accuracy of the prediction has a great impact on the protocol's ability to select a more stable route.

Though less stable routes are more likely to be selected when predictions are less accurate for further away in time, the results have demonstrated that link duration predictions with lower accuracy can still be useful in enabling smoother route transitions. This is because the route updates are used in AODV-PU to perform route maintenance. This allows the source to be informed about the latest predicted route duration, which enables the source to re-establish the route before it becomes unavailable. This can be achieved because route duration updates do not require to make predictions that are too far into the future, hence the predictions are more accurate.

The studies in this thesis have provided some insights about the trade-offs when using predictions. One trade-off is the extended initial route establishment time at route set up, which does not guarantee a more stable route to be found if the accuracy of the prediction is not guaranteed. So perhaps using prediction to find a more stable route at route setup may not be worth doing, given that the trade-off and the reward is not balanced. Another trade-off related to the fore-mentioned trade-off is that the longer end-to-end delay time due to longer paths with more hops were more likely to be predicted to be more stable. Once again however, a more stable route cannot be guaranteed if the prediction is not guaranteed to be accurate.

Through the studies in this thesis, it can be concluded that perhaps prediction may not be as useful for selecting a more stable route during route establishment phase as first anticipated. This is because this requires accurate predictions to be made well in advance, and it is very difficult to make predictions for something further away in the future, as the accuracy of prediction drops as the time gap increases. However, it is found that prediction may be more useful for maintaining and re-establishing an alternative route, since these only require accurate predictions to be made a few seconds in advance, as opposed to tens of minutes or hours in advance may be needed for initial route selections. Hence, further research can be performed in using prediction for route maintenance and route re-establishment.

For further study, it would be interesting to see how well other traffic types (including traffic over TCP) perform over AODV-PP and AODV-PU. Because with these protocols, route re-establishments are often accomplished before link loss, which reduces packet loss and route re-establishment delay substantially, TCP may work better over these protocols compared to AODV. Furthermore, stable routes can be established with AODV-PP, thus it is expected that TCP may function better over AODV-PP than AODV-PU.

One possible area of future research may be to perform local repair ahead of time for route maintenance if a link is predicted to break. However, using local repair may reduce the chance of finding a more stable alternative route from the source to destination if the prediction is more reliable.

Another possible way to move forward could be to allow the routing protocol to maintain multiple paths, since multiple paths were already found during route setup when prediction is used. Having multiple paths can enable a smooth transition to be made to an alternative route when needed. However, more routing traffic may be needed to maintain multiple paths.

Although selecting a stable route cannot be guaranteed when the accuracy of prediction well in advance is not guaranteed, that is not to say that prediction is useless for route establishment. Prediction may still be useful for route establishment if the accuracy or the reliability of the prediction can be evaluated and passed on with RREQ during route setup. For example, there may be times when there are definite stable routes available, and if such routes can be found and used, it would be much better than maintaining the less stable routes. Furthermore, the route establishment time may be shortened by allowing intermediate nodes to send a route reply upon receiving a route request if it has a route to the destination, however, its trade-off is having a reduced chance of finding a more stable route. With low prediction accuracy, allowing intermediate nodes to send a route reply upon receiving a route request could be a good option.

To counter some of these trade-offs, it may also be possible to use some adaptive methods so that for example if the prediction is estimated to be more accurate, the routing protocol can use prediction for route establishment. Alternatively, if the prediction is estimated to be not so reliable, standard shortest path route selection can still be used.

This thesis has been focused on AODV. However, predictions can be implemented on other routing protocols. Hence, expanding the use of prediction on other routing protocols may be left for future work.

Depending on what the user requirements are: if the requirement is to have short end-to-end delay for good voice conversations, protocols such as AODV or DSR may be good enough; if the requirement is to have higher route stability for TCP traffic, some accurate predictions may be useful; if the requirement is to minimise packet loss, prediction may be used to assist in setting up for a smooth route transition; or if the requirement is to have all of the above, then unfortunately, no MANET routing protocol to date can satisfy all these requirements at once for all situation without conditions. Overall, there is still no perfect routing protocol suitable for all types of MANETs for all cases. Even with prediction, it is not necessarily better than the basic routing protocols such as AODV, DSR and OLSR.

Appendix A

AODV Model Corrections

The changes that have been made to fix OPNET's AODV model are included in this appendix.

A.1 Send RERR

According to AODV standard, RERR packets should be broadcast if there are one or more precursors destined to one or more destinations. In OPNET's AODV model, the RERR packets may sometimes be unicasted when it should be broadcast. This is because there is one case missing in the code for setting the destination address for the RERR packet. This can be fixed by adding several lines of code to the aodv_rte_route_error_process() function in the Function Block of the aodv_rte.m model, as indicated in the pseudo code below. Note that lines 24 to 27 and 51 to 54 of the pseudo code are added to enable RERR packets to be broadcast to all the precursors when required.

Code A.1: aodv_rte_route_error_process() Function

```
void aodv_rte_route_error_process (neighbour_addr, unreachable_nodes_list) {
  ...
  create unreachable_dests_list;
  set Got_One_Precursor to FALSE;
  ...
  if (link_break_detected for neighbour_addr) {
    for (each route_entry in the routing table) {
      if (neighbour_addr is used to reach the destination of the route_entry) {
        set num_precursors to the size of (route_entry->precursor_list);
        if (num_precursors > 0) {
          create unreachable_node with destination and its sequence_number;
          add unreachable_node to unreachable_dests_list;
          if (num_precursors == 1 && !Got_One_Precursor) {
            remove precursor_address in route_entry->precursor_list;
            set unique_precursor_addr to precursor_address;
              /* unique_precursor_addr will be the dest address for RERR */
            set Got_One_Precursor to TRUE ;
          }
          else if (num_precursors > 1) {
            set unique_precursor_addr to INETC_ADDRESS_INVALID;
            while (size of (route_entry->precursor_list) > 0)
              remove precursor_address in route_entry->precursor_list;
          }
          else if (num_precursors == 1 && Got_One_Precursor) {
            remove precursor in route_entry->precursor_list;
            set unique_precursor_addr to INETC_ADDRESS_INVALID;
          }
        }
      }
    }
  }
  if (rerr_received) {
    for (each route_entry in the routing table) {
      for (each unreachable_node in unreachable_nodes_list) {
        if (route_entry->destination is unreachable_node->destination &&
            route_entry->next_hop_addr is neighbour_addr) {
          set num_precursors to the size of (route_entry->precursor_list);
          if (num_precursors > 0) {
            create unreachable_node with destination and its sequence_number;
            add unreachable_node to unreachable_dests_list;
            if (num_precursors == 1 && !Got_One_Precursor) {
              remove precursor_address in route_entry->precursor_list;
              set unique_precursor_addr to precursor_address;
              set Got_One_Precursor to TRUE ;
            }
            else if (num_precursors > 1) {
              set unique_precursor_addr to INETC_ADDRESS_INVALID;
              while (size of (route_entry->precursor_list) > 0)
                remove precursor_address in route_entry->precursor_list;
            }
            else if (num_precursors == 1 && Got_One_Precursor) {
              remove precursor_address in route_entry->precursor_list;
              set unique_precursor_addr to INETC_ADDRESS_INVALID;
            }
          }
        }
      }
    }
  }
}
```

A.2 Precursor List Maintenance

The problem with maintaining a correct precursor list is caused because there is no code in OPNET to add the precursor node to the precursor list upon receiving a data packet. This problem can be fixed by adding the code from line 28 to line 39 of the aodv_rte_data_routes_expiry_time_update() function illustrated in Code A.2 below.

Code A.2: aodv_rte_data_routes_expiry_time_update() Function

```
1  void aodv_rte_data_routes_expiry_time_update () {
2    ...
3    call aodv_route_table_entry_get (route_table, src_addr) to get src_route_entry;
4    call aodv_route_table_entry_get (route_table, dest_addr) to get dest_route_entry;
5
6    if (src_route_entry exists) {
7      get the prev_hop_addr of pkptr;
8      call aodv_rte_neighbor_connectivity_table_update (prev_hop_addr,...)  to set
9          neighbor connectivity expiry time = curr_time + ACTIVE_ROUTE_TIMEOUT;
10
11     if (src_route_entry->next_hop_addr is prev_hop_addr) {
12       call aodv_route_table_entry_expiry_time_update() to set
13           src_route_entry->route_expiry_time = curr_time + ACTIVE_ROUTE_TIMEOUT;
14     }
15
16     call aodv_route_table_entry_get (route_table, prev_hop_addr) to get
17         the prev_route_entry from the routing table;
18     if (prev_route_entry != OPC_NIL && src_addr is not prev_hop_addr &&
19         prev_route_entry->next_hop_addr is prev_hop_addr)
20       call aodv_route_table_entry_expiry_time_update() to set
```

```
        prev_route_entry->route_expiry_time = current_time + ACTIVE_ROUTE_TIMEOUT;
  }

  if (dest_route_entry exists) {
    update route_entry(s) as normal;
  }

  if (this node is not the source of the packet && src_route_entry does not exist) {
    call aodv_route_table_entry_create() to create a new route_entry for the source;
  }

  if (src_route_entry exists && dest_route_entry exists) {
    call aodv_rte_neighbor_connectivity_table_update (prev_hop_addr,...)  to set
        neighbor connectivity expiry time = curr_time + ACTIVE_ROUTE_TIMEOUT;

    /* Update the precursor lists */
    add precursor (next_hop_addr) to the src_route_entry;
    add precursor (prev_hop_addr) to the dest_route_entry;
  }
}
```

A.3 Route Expiry Time Update

Lines 15 to 22 are added to the aodv_rte_route_table_entry_update() function to prevent early link timeout.

Code A.3: aodv_rte_route_table_entry_update() Function

```
1  void aodv_rte_route_table_entry_update () {
2    ...
3    if (route_entry does not exist) {
4      ...
5    }
6    else {
7      if ((hello_msg) ||
8        (sequence_num > dest_seq_num) ||
9        ((sequence_num == dest_seq_num) && (route_entry_state == AodvC_Invalid_Route)) ||
10       ((sequence_num == dest_seq_num) && (hop_count < 2))) {
11         ...
12     }
13
14     // The following is added to extend the timeout period.
15     for each route_entry in the routing table {
16       check if the route is valid;
```

```
      if (route_entry->next_hop_addr is ip_packet->prev_hop_addr &&
          route_entry->route_expiry_time - curr_time < ACTIVE_ROUTE_TIMEOUT) {
        call aodv_route_table_entry_expiry_time_update() to set
            route_entry->route_expiry_time = curr_time + ACTIVE_ROUTE_TIMEOUT;
      }
    }
  }
}
```

A.4 Neighbour Update

Changes for neighbour updates need to be made to two functions. One is when a HELLO packet is received, and the other is when a data packet is received.

A.4.1 HELLO Packet

Upon receiving of a HELLO packet, the route expiry time for all the destinations that are using that link needs to be updated, not just for the neighbour. Therefore, the solution to this problem needs to be added to the aodv_rte_route_table_entry_from_hello_update() function as indicated on lines 5 and 6 of the pseudo code below.

Code A.4: aodv_rte_route_table_entry_from_hello_update() Function

```
1 void aodv_rte_route_table_entry_from_hello_update () {
2   ...
3   if (route_entry to the previous hop exists) {
4     update the sequence number;
5     call aodv_rte_route_table_entry_update() to update
6         route_entry->route_expiry_time of all relevant route entries;
7   }
8 }
```

A.4.2 Data Packet

Upon receiving of a data or a routing packet, all neigbouring nodes' connectivity expiry times should be updated to maintain the connectivity of the neighbouring nodes that are in use. Thus, the following code (lines 7 to 9) is added to the aodv_rte_data_routes_expiry_time_update() function in Code A.2 on page 136 to achieve this purpose. Also note that in OPNET's code, aodv_route_table_entry_param_get() is called to get the next hop address of the source route entry. This is inappropriate, because it is possible that the next hop address in the source route entry is different to the previous node of the received data packet's path. Thus, line 7 is used to get the previous hop address from the received data packet, instead of aodv_route_table_entry_param_get() was used in OPNET's code. Furthermore, lines 11 and 19 are added to ensure that the correct route entries are updated. Line 11 is added to ensure that the source route entry is only updated if the next hop address of the source route entry is the same as the previous hop address of the received packet. Similarly, the condition in line 19 is added for the route entry of the previous hop.

Appendix B

Modifications to AODV for AODV-PP

The modifications that have been made to AODV to develop AODV-PP are included in this appendix. Note that the pseudo codes in this appendix are not the only modifications made to the OPNET's model. There may be some minor modifications that are not mentioned.

B.1 Packet Structures

This section shows all the packet structures that have been changed and used in AODV-PP.

B.1.1 RREQ Packet Structure

In AODV-PP, routes are selected based on the predicted route expiry time of the route, called the "Pathchange Time". This parameter will have to be updated and carried with the RREQ packet through the network for route setup. Therefore, an extra field is added to the standard AODV's RREQ packet, as illustrated in Table B.1 below. In additional to this, the first bit of the "Reserved" field is replaced by the "Pre-Route" flag (i.e. after the "Unknown sequence number" flag). This flag is to indicate that the RREQ is sent while a route was still valid.

00	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Packet Type								Flags						Reserved										Hop Count							
RREQ ID																															
Destination IP Address																															
Destination Sequence Number																															
Source IP Address																															
Source Sequence Number																															
Pathchange Time																															

Table B.1: *RREQ Packet Structure.*

Since the "Pathchange Time" is the predicted expiry time of the path, which is equivalent to the "route expiry time" that have been used in the route entries of OPNET's AODV model. For simplicity, "Pathchange Time" with type "double" is used. Initially, the "Pathchange Time" is set to the maximum value, which is infinity in this case. Alternatively, "Pathchange Time" can be replaced by "Path Lifetime", which represents the predicted lifetime of the route.

B.1.2 RREP Packet Structures

According to standard AODV, the "Lifetime" field of the RREP packet structure is set to a fixed duration that a route is expected to last. In AODV-PP, this "Lifetime" field is set to the route expiry time of the route to the source. Another difference is that AODV-PP has an additional field called "Pathchange Time". Similar to the RREQ packet, the "Pathchange Time" field stores the predicted route expiry from the destination, and is updated at each hop along the return route. The modified RREP packet is shown in Table B.2.

00	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Packet Type								Flags		Reserved									Prefix					Hop Count							
Destination IP Address																															
Destination Sequence Number																															
Source IP Address																															
Lifetime (optional)																															
Pathchange Time																															

Table B.2: *RREP Packet Structure.*

Again, the "Pathchange Time" field is initially set to the largest value by the destination, which is infinity for OPNET. After the linkchange_time is predicted at each hop, the node will set or update the "pathchange_time" of the route entry, and the "Pathchange Time" field of the RREP will also be updated and sent to the next hop towards the source.

B.2 Process RREQ Packets

Upon receiving a RREQ packet, it is processed in the modified function, aodv_rte_rreq_pkt_arrival_handle(), as represented in pseudo Code B.2 below.

Code B.2: aodv_rte_rreq_pkt_arrival_handle() Function

```
void aodv_rte_rreq_pkt_arrival_handle (... ,
                    AodvT_Packet_Option* tlv_options_ptr) {
  set rreq = (AodvT_Rreq*) tlv_options_ptr->value_ptr;
  ...
  /* Added for Prediction to predict the pathchange time */
  if (predictive_aodv) {
    call predict_linkchange_time (prev_hop_addr) to predict linkchange_time;
    if (linkchange_time < curr_time) {
      linkchange_time = curr_time;
    }
    pathchange_time = min (rreq->pathchange_time, linkchange_time);
    rreq->pathchange_time = pathchange_time;
  }
  /* Prediction Ends */

  call aodv_rte_neighbor_connectivity_table_update (prev_hop, FALSE) to
      add or update the neighbour connectivity table;
```

```

/* Modified for Prediction to update the route entry to prev_hop */
if (predictive_aodv) {
  /* Update the route entry for AODV-PP/AODV-PU */
  if (rreq->pre_route_flag is TRUE)
    call aodv_rte_route_table_entry_update();
  else
    call aodv_rte_route_table_entry_update();
}
else {
  /* Update the route entry for AODV */
  call aodv_rte_route_table_entry_update();
}
/* Prediction Modification Ends */

...

/* Modified for Prediction to decide whether to accept */
/* or reject the RREQ by checking the RREQ forwarded.  */
call aodv_route_request_forward_entry_exists () to obtain
    the existing route_entry->pathchange_time and to
    check if a RREQ has already been received;
if (route_request_forward_entry_exists) {
/* Prediction Modification Ends */
  if (standard_aodv) {
    discard RREQ/IP Pkt;
    return;
  }
  /* Added for Prediction */
  else if (predictive_aodv) {
    if (pathchange_time < route_entry->pathchange_time) {
      discard RREQ/IP Pkt;
      return;
    }
  }
  /* Prediction Ends */
}

/* Added for Prediction to discard RREQ if the route's */
/* pathchange time is too short.                       */
if (predictive_aodv is TRUE && pathchange_time < curr_time) {
  discard RREQ/IP Pkt;
  return;
}
/* Prediction Ends */

/* Modified for Prediction to add new RREQ in the request table. */
call aodv_request_table_forward_rreq_insert (..., pathchange_time);
/* Prediction Modification Ends */

```

```
rreq->hop_count++;
if (rreq->hop_count > net_diameter) {
  discard RREQ/IP Pkt;
  return;
}

if (standard_aodv)
  set min_lifetime = ((2 * net_traversal_time) -
                      (2 * rreq->hop_count * node_traversal_time));
/* Added for Prediction to calculate the lifetime different */
/* to AODV.                                                 */
else if (predictive_aodv)
  set min_lifetime = rreq->pathchange_time - op_sim_time();
/* Prediction Ends */

call aodv_route_table_entry_get() to get source's route_entry;
if (route_entry does not exist) {
  /* Modified for Prediction to create a route entry, */
  /* as new input parameters are added to the         */
  /* aodv_route_table_entry_create() function.        */
  call aodv_route_table_entry_create(..., lifetime, ..., Aodv_Stage) to
      create and add route_entry to route_table;
  /* Prediction Modification Ends */
  call aodv_rte_all_pkts_to_dest_send() to send all queued packets to
      the source of the RREQ;
}
else if (standard_aodv &&
          ((rreq->src_seq_num > route_entry->dest_seq_num) ||
           (rreq->src_seq_num == route_entry->dest_seq_num &&
            rreq->hop_count < route_entry->hop_count) ||
           (rreq->src_seq_num == route_entry->dest_seq_num &&
            route_entry->route_entry_state is Invalid)
           )
         ) {
  update route_entry as normal;
}
/* Added for Prediction */
else if (predictive_aodv) {
  set pre_route_expiry_time = curr_time + 2.0 * node_traversal_time *
                               (ttl - 1 + timeout_buffer);

  if ((pre_route_expiry_time < rreq->pathchange_time - 0.1) &&
      ((route_entry->route_entry_state is not Valid) ||
       (rreq->src_seq_num > route_entry->pre_route_dest_seq_num) ||
       ((rreq->src_seq_num == route_entry->pre_route_dest_seq_num) ||
        (rreq->pathchange_time >=
            route_entry->pre_route_route_expiry_time + 0.1)))) {
    update route_entry for AODV-PP/AODV-PU as follow:
      set route_entry->pre_route_dest_seq_num = rreq->src_seq_num;
      set route_entry->pre_route_next_hop_addr = prev_hop_addr;
```

```
        set route_entry->pre_route_next_hop_port_info = in_port_info;
        set route_entry->pre_route_hop_count = rreq->hop_count;
        set route_entry->pre_route_route_expiry_time = rreq->pathchange_time;
        set route_entry_ptr->pre_route_state = TRUE;
        call aodv_route_table_entry_pre_route_timer_update() to set an interrupt
            at route_entry->pre_route_expiry_timer = pre_route_expiry_time;

        if (route_entry->route_entry_state is not Valid) {
          calculate route duration stats;
          set route_entry->dest_seq_num = rreq->src_seq_num;
          set route_entry->next_hop_port_info = in_port_info;
          set route_entry->hop_count = rreq->hop_count;

          if (route_entry->next_hop_addr is not prev_hop_addr)
            call aodv_route_table_entry_next_hop_update() to update
                the route_entry with the previous hop's information;
          call aodv_route_table_entry_expiry_time_update() to
              set route_entry->route_expiry_time = rreq->pathchange_time;
          call aodv_route_table_entry_state_set() to
              set route_entry->route_entry_state = Valid;
          call aodv_rte_all_pkts_to_dest_send() to
              send all queued application packets to the
              destination using the new route;
        }

        /* Set the valid sequence number flag field. */
        if (rreq->src_seq_num == AODVC_DEST_SEQ_NUM_INVALID)
          route_entry->pre_route_valid_dest_sequence_number_flag = FALSE;
        else
          route_entry->pre_route_valid_dest_sequence_number_flag = TRUE;
    }
    else if (route_entry->pre_route_next_hop_addr != prev_hop_addr) {
      discard RREQ/IP Pkt;
      return;
    }
    else if (pre_route_expiry_time > rreq->pathchange_time) {
      discard RREQ/IP Pkt;
      return;
    }
  }
  /* Prediction Ends */
  else if (route_entry->next_hop_addr != prev_hop_addr) {
    discard RREQ/IP Pkt;
    return;
  }

  if (this_node is dest_node) {
    /* If this node is the destination of the RREQ. */
    if (standard_aodv) {
      /* If AODV, send RREP immediately. */
```

```
        send RREP now;
      }
      else {
        /* Added for Prediction to set a timer to send RREP. */
        set route_entry->rrep_send_timer
            to send RREP;
        call aodv_route_table_rrep_send_timer_update() to send RREP
            at route_entry->rrep_send_timer;
        /* Prediction Ends */
      }

      discard RREQ/IP Pkt;
      return;
    }

    if (gateway is enabled &&
        dest is a known-available external destination) {
      ...
    }

    if (dest_route_entry exists) {
      if (dest_route_entry->dest_seq_num >= rreq->dest_seq_num &&
          dest_route_entry->route_entry_state is Valid) {
        if (!rreq->dest_only) {
          /* This is for Non Destination Only, */
          /* This option is not used.          */
        }
      }
    }
    else if (gateway is enabled) {
      /* This option is not used. */
    }

    if (ttl == 1) {
      discard RREQ/IP Pkt;
      return;
    }

    set new_ttl = ttl -1;
    encapsulate RREQ into an IP packet;

    add statistics;

    schedule to send RREQ;
}
```

The previous function, aodv_rte_rreq_pkt_arrival_handle(), calls other functions to complete the process upon receiving a RREQ. The required functions that have been modified or added include:

1. predict_linkchange_time() (Code B.2.1);
2. aodv_request_table_forward_rreq_insert() (Code B.2.2);
3. aodv_route_request_forward_entry_exists() (Code B.2.3);
4. aodv_rte_route_table_entry_update() (Code B.2.4);
5. aodv_route_table_entry_create() (Code B.2.5);
6. aodv_route_table_entry_next_hop_update() (Code B.2.6);
7. aodv_route_table_rrep_send_timer_update() (Code B.2.7); and
8. aodv_route_table_pre_route_expiry_timer_update() (Code B.2.8).

B.2.1 predict_linkchange_time Function

The function, predict_linkchange_time(), is a new function that is called to get the predicted linkchange time for a wireless link to a specific neighbouring node. Thus, this function requires the neighbouring node's address as its input, and it returns the predicted link change time.

Code B.2.1: predict_linkchange_time() Function

```
double predict_linkchange_time (neighbour_addr) {
  get neighbour_info;

  /* For each node in the network, find the neighbour */
  for (i = 0; i < track_info_ptr->n_other_nodes; i++) {
    get the track_info for node i;
    if (node i is the neighbour) {
      call op_ima_obj_pos_get_time() to get the
          coordinates (y11, z11) of this node;
      call op_ima_obj_pos_get_time() to get the
          coordinates (y21, z21) of the neighbour;
      calculate distance = sqrt((y21-y11)*(y21-y11) + (z21-z11)*(z21-z11));

      if (predict_method is PERFECT_ALL) {
        /* This part is for perfect prediction */
        get linkchange_time = others_info->pred_linkchange_time;
        set link_duration = linkchange_time - curr_time;
      }
      else if (predict_method is LET) {
        /* This part is for predicting using LET */
        call predict_link_break_time_let (tx_range, self_objid, neighbour_objid)
            to predict the linkchange_time;
        set link_duration = linkchange_time - curr_time;
      }

      if ((link_duration > 0) && (distance < tx_range)) {
        /* If link is still available, return the predicted linkchange_time */
        neighbour_info->link_duration = link_duration;
        neighbour_info->linkchange_time = linkchange_time;
        call inet_addr_hash_table_item_insert() to add the neighbouring node
            to keep track of its neighbours;

        return (linkchange_time);
      }
      else {
        /* If link is not available, return the predicted linkchange_time */
        print a warning message;

        return (linkchange_time);
        }
    }
  }

  return -1;
}
```

B.2.2 aodv_request_table_forward_rreq_insert Function

The function, aodv_request_table_forward_rreq_insert(), is used to insert the RREQ packet information into a table, in order to keep track of the RREQ packets that have been received.

Code B.2.2: aodv_request_table_forward_rreq_insert() Function

```
void aodv_request_table_forward_rreq_insert (req_table, req_id,
                    originator_addr, pathchange_time) {
  /* Check if there exists a req_entry_list for the originator_addr */
  call inet_addr_hash_table_item_get() to get req_entry_list;

  if (req_entry_list does not exists) {
    call op_prg_list_create () to create req_entry_list;
    call inet_addr_hash_table_item_insert () to add
          req_entry_list into req_table->forward_request_table;
  }

  /* Create an entry for this new request */
  req_entry = aodv_request_table_forward_entry_mem_alloc ();
  req_entry->request_id = req_id;
  req_entry->pathchange_time = pathchange_time;
  req_entry->insert_time = curr_time;

  /* Insert this new request into the request table */
  call op_prg_list_insert (req_entry_list, req_entry, OPC_LISTPOS_TAIL)
      to add req_entry to the end of req_entry_list;

  orig_addr_ptr = inet_address_copy_dynamic (&originator_addr);
  call op_intrpt_schedule_call () to schedule an interrupt
      at curr_time + req_table->forward_request_expiry_time
      to clear the req_entry_list;
}
```

B.2.3 aodv_route_request_forward_entry_exists Function

The aodv_route_request_forward_entry_exists() checks if the node has received a RREQ packet with the same request ID through a more stable route before by looking into the route request entry table. This function also obtains the best pathchange time of the route request entry before this new RREQ packet arrived.

Code B.2.3: aodv_route_request_forward_entry_exists() Function

```
Boolean aodv_route_request_forward_entry_exists (req_table, req_id,
      originator_addr, incoming_pathchange_time, *existing_pathchange_time) {
  /* Check if there exists a req_entry_list for the originator_addr */
  call inet_addr_hash_table_item_get() to get the
      req_entry_list of the originator_addr from req_table;

  if (req_entry_list does not exists) {
    return FALSE;
  }

  set list_size = op_prg_list_size (req_entry_list);

  for (n = 0; n < list_size; n++) {
    call op_prg_list_access (req_entry_list, n) to get the n-th
```

```
        req_entry of the req_entry_list;

    if (req_entry->request_id == req_id) {
      set *existing_pathchange_time = req_entry->pathchange_time;

      if (incoming_pathchange_time > req_entry->pathchange_time)
        set req_entry->pathchange_time = incoming_pathchange_time;

    return TRUE;
    }
  }

  if (req_entry_list is Empty) {
    call inet_addr_hash_table_item_remove() to remove the
        originator_addr's req_entry_list from the req_table;
    call prg_mem_free (req_entry_list) to free req_entry_list;
  }

  return FALSE;
}
```

B.2.4 aodv_rte_route_table_entry_update Function

The aodv_rte_route_table_entry_update() function is called to create or update the route_entry to the previous hop (prev_hop) of the incoming control packet. This function is called by the aodv_rte_rreq_pkt_arrival_handle(), aodv_rte_rrep_pkt_arrival_handle() and aodv_rte_route_table_entry_from_hello_update() functions.

Code B.2.4: aodv_rte_route_table_entry_update() Function

```
void aodv_rte_route_table_entry_update (...,
          AodvT_Packet_Option* tlv_options_ptr, Aodv_Stage stage) {
  get prev_hop from the arrived packet;
  call aodv_route_table_entry_get () to get prev_hop's route_entry;
  if (tlv_options_ptr->type is AODVC_ROUTE_REQUEST) {
    set rreq = (AodvT_Rreq*) tlv_options_ptr->value_ptr;
    ... (set sequence_num and lifetime)
    /* Added for Prediction */
    if (predictive_aodv) {
      set lifetime = predicted_linkchange_time;
      if (route_entry exists && rreq->pre_route_flag &&
          !manet_rte_address_belongs_to_node (..., rreq->dest_addr)) {
```

```
        route_entry->pre_route_state = rreq->pre_route_flag;
      }
    }
    /* Prediction Ends */
  }
  else if (tlv_options_ptr->type is AODVC_ROUTE_REPLY) {
    set rrep = (AodvT_Rrep*) tlv_options_ptr->value_ptr;
    ... (set sequence_num and lifetime)
    /* Added for Prediction */
    if (predictive_aodv) {
      set lifetime = predicted_linkchange_time;
    }
    /* Prediction Ends */
  }
  else if (tlv_options_ptr->type is AODVC_HELLO) {
    ... (set sequence_num and lifetime)
  }
  ...
  if (route_entry does not exist) {
    if (standard_aodv) {
      ...
    }
    /* Added for Prediction */
    else if (predictive_aodv) {
      if (tlv_options_ptr->type is AODVC_ROUTE_REQUEST) {
        call aodv_route_table_entry_create  (..., lifetime, ...,
            ttl, aodv_type, aodv_stage) to create and add
            route_entry for prev_hop to route_table;
      }
      else if (tlv_options_ptr->type is AODVC_ROUTE_REPLY) {
        call aodv_route_table_entry_create  (..., lifetime, ...,
            ..., aodv_type, aodv_stage) to create and add
            route_entry for prev_hop to route_table;
      }
      else if (tlv_options_ptr->type is AODVC_HELLO) {
        call aodv_route_table_entry_create  (..., lifetime, ...,
            1, aodv_type, aodv_stage) to create and add
            route_entry for prev_hop to route_table;
      }
    }
    /* Prediction Ends */
  }
  else if (standard_aodv) {
    update route_entry;
    ...
  }
  else if (predictive_aodv) {
    /* Added for Prediction so that the right variables */
    /* of the route_entry is updated correctly.         */
    copy the following parameters from route_entry:
        dest_seq_num = route_entry->dest_seq_num;
        hop_count = route_entry->hop_count;
        route_entry_state = route_entry->route_entry_state;
        pre_route_dest_seq_num = route_entry->pre_route_dest_seq_num;
        pre_route_hop_count = route_entry->pre_route_hop_count;
        pre_route_state = route_entry->pre_route_state;

      set pre_route_expiry_time = curr_time;
      if (tlv_options_ptr->type is AODVC_ROUTE_REQUEST) {
        set pre_route_expiry_time += 2.0 * node_traversal_time *
                                    (ttl - 1 + timeout_buffer);
      }

      if ((tlv_options_ptr->type is AODVC_HELLO) ||
          (route_entry_state is not Valid) ||
          ((lifetime > pre_route_expiry_time) &&
           ((tlv_options_ptr->type == AODVC_ROUTE_REPLY) &&
            (route_entry->next_hop_addr is prev_hop) &&
            (sequence_num > dest_seq_num)) ||
           ((tlv_options_ptr->type == AODVC_ROUTE_REPLY) &&
            (sequence_num >= dest_seq_num) &&
            (rrep->pathchange_time > route_entry->route_expiry_time)) ||
           ((!pre_route_state) &&
            (sequence_num > dest_seq_num)) ||
          (lifetime > route_entry->pre_route_route_expiry_time + 0.1)
          )) {
        if (route_entry_state is not VALID) {
          call aodv_route_table_entry_next_hop_update() to update
              the route_entry with the new next hop's information:
        }
      }
      /* Prediction Ends */
    }
}
```

Where aodv_type is either AODV or PREDICT, and aodv_stage keeps track of what stage of AODV the model is in, such as Receiving RREQ, Receiving PreRREQ or Receiving RREP.

B.2.5 aodv_route_table_entry_create Function

The aodv_route_table_entry_create() function creates a route entry (route_entry) in the route table (route_table). It is called when there is no route entry exists.

This function is modified so that it copies the route information into the additional pre-route variables in the route_entry.

Code B.2.5: aodv_route_table_entry_create() Function

```
AodvT_Route_Entry* aodv_route_table_entry_create (route_table, dest_addr,
      subnet_mask, next_hop_addr, out_port_info, num_hops, dest_seq_num,
      expiry_time, link_expiry_time, ttl, aodv_type, aodv_stage) {
  allocate memory for route_entry;
  call ip_cmn_rte_table_dest_prefix_create (dest_addr, subnet_mask) to
      set route_entry->dest_prefix;   // this is required in OPNET's AODV.
  set route_entry->dest_seq_num = dest_seq_num;
  if (dest_seq_num is Valid)
    set route_entry->valid_dest_sequence_number_flag to TRUE;
  else
    set route_entry->valid_dest_sequence_number_flag to FALSE;
  set route_entry->route_entry_state = AodvC_Valid_Route;
  set route_entry->next_hop_addr = next_hop_addr;
  set route_entry->next_hop_port_info = out_port_info;
  set route_entry->hop_count = num_hops;
  set route_entry->route_expiry_time = curr_time + expiry_time;
  set route_entry->route_request_time = curr_time;
  set route_entry->route_begins_time = curr_time;
  set route_entry->rrep_send_timer_enabled to FALSE;
  set route_entry->rrep_send_timer = 0;
  allocate memory for route_entry->rreq_option_ptr;
  set route_entry->ip_dgram_fd_ptr to NULL;
  if (predictive_aodv) {
    set route_entry->route_expiry_time = expiry_time;
    set route_entry->pre_route_start_timer = route_entry->route_expiry_time -
              2.0 * route_entry->node_traversal_time *
              (route_entry->net_diameter + route_entry->timeout_buffer);

    set route_entry->pre_route_expiry_timer = 0;
    if (aodv_stage == PRER_RREQ || aodv_stage == PRED_RREQ) {
      set route_entry->pre_route_expiry_timer = curr_time +
              2.0 * route_entry->node_traversal_time *
              (ttl - 1 + route_table->timeout_buffer);
      call op_intrpt_schedule_call() to set an interrupt at
          route_entry->pre_route_expiry_timer;
    }
  }
  else {
    set route_entry->pre_route_start_timer = -1;
    set route_entry->pre_route_expiry_timer = -1;
  }

  set route_entry->route_expiry_update_lptr = op_prg_list_create ();
  set route_entry->last_route_expiry_update_time = 0;
  set route_entry->route_expiry_update_timer = 0;
  set route_entry->new_route_expiry_time = set route_entry->route_expiry_time;
  set route_entry->link_expiry_time = link_expiry_time;
  set route_entry->new_link_expiry_time = link_expiry_time;
  set route_entry->is_shortest_link_duration to FALSE;
  if (abs(route_entry->route_expiry_time - route_entry->link_expiry_time) < 0.001)
    set route_entry->is_shortest_link_duration to TRUE;

  set route_entry->pre_route_dest_seq_num = dest_seq_num;
  set route_entry->pre_route_next_hop_addr = next_hop_addr;
  set route_entry->pre_route_next_hop_port_info = out_port_info;
  set route_entry->pre_route_hop_count = num_hops;
  set route_entry->pre_route_cost = 0;
  set route_entry->pre_route_route_expiry_time = expiry_time;
  set route_entry->pre_route_link_expiry_time = link_expiry_time;
  if (aodv_stage == PRER_RREQ || aodv_stage == PRED_RREQ) {
    set route_entry->pre_route_state to TRUE;
  }
  else {
    set route_entry->pre_route_state to FALSE;
  }
  set route_entry->pre_route_ready to FALSE;
  set route_entry->is_source_node to FALSE;
  set route_entry->last_app_pkt_to_dest_time = -1;
  set route_entry->last_pkt_to_dest_time = -1;
  if (route_entry->route_expiry_time > curr_time) {
    if (expiry_time != INFINITY) {
      call op_intrpt_schedule_call() to set an interrupt at
          route_entry->route_expiry_time;
    }
  }
  else {
    set route_entry->route_entry_state = AodvC_Invalid_Route;
    FRET (route_entry_ptr);
  }

  call inet_addr_hash_table_item_insert() to insert
      route_entry into AODV route_table;
  call Inet_Cmn_Rte_Table_Entry_Add_Options() to insert
      route in the IP routing table

  increment route_entry->active_route_count;

  update statistics;

  return route_entry;
}
```

B.2.6 aodv_route_table_entry_next_hop_update Function

The aodv_route_table_entry_next_hop_update() function is called by the aodv_rte_rreq_pkt_arrival_handle(), aodv_rte_rrep_pkt_arrival_handle() and aodv_rte_route_table_entry_update() functions. This is to update the route_entry with the new sequence number of the destination (sequence_num), next hop address to the destination (next_hop_addr), hop count to the destination (hop_count), and out port information (out_port_info).

Code B.2.6: aodv_route_table_entry_next_hop_update() Function

```
void aodv_route_table_entry_next_hop_update (route_table, route_entry,
      sequence_num, next_hop_addr, hop_count, out_port_info) {
  if ((route_entry->route_entry_state is valid) &&
      (inet_cmn_rte_table_entry_exists (route_table->ip_cmn_rte_table_ptr,
          route_entry->dest_prefix, &entry_ptr) is OPC_COMPCODE_SUCCESS)) {
    call Inet_Cmn_Rte_Table_Entry_Update() to update the routing table;
  }
  set route_entry->dest_seq_num = sequence_num;
  set route_entry->next_hop_addr = next_hop_addr;
  set route_entry->next_hop_port_info = in_port_info;
  set route_entry->hop_count = hop_count;
  set route_entry->route_request_time = curr_time;
  set route_entry->pre_route_dest_seq_num = sequence_num;
  set route_entry->pre_route_next_hop_addr = next_hop_addr;
  set route_entry->pre_route_next_hop_port_info = in_port_info;
  set route_entry->pre_route_hop_count = hop_count;
}
```

B.2.7 aodv_route_table_rrep_send_timer_update Function

The aodv_route_table_rrep_send_timer_update() function is called by the aodv_rte_rreq_pkt_arrival_handle() at the destination node to set a timer to send a RREP back to the source. A RREP is not sent immediately because there may be other RREQs arrive later that have a better route lifetime. This is to avoid nodes receiving multiple RREPs making the source and some intermediate nodes to reset the route multiple times in a very short period of time. Changing routes quickly may cause problems in TCP.

Code B.2.7: aodv_route_table_rrep_send_timer_update() Function

```
int aodv_route_table_rrep_send_timer_update (
                    route_entry, dest_addr, exp_time, code) {

  cancel existing event(s) in route_entry->rrep_send_timer_evhandle;

  if (curr_time <= exp_time && exp_time < OPC_DBL_INFINITY) {
    /* If the exp_time is between now and infinity, set an */
    /* interrupt to send RREP when the timer expires.      */
    call op_intrpt_schedule_call (exp_time, code, aodv_rte_rrep_send_timer_handle,
        route_entry) to schedule an interrupt call at rrep_send_timer to send a
        RREP, and keep the event handler in route_entry->rrep_send_timer_evhandle;
    set route_entry->rrep_send_timer_enabled = TRUE;
  }
  else {
    /* Do not schedule any event if the exp_time is */
    /* before the current time or if it is infinity.*/
    if (exp_time < curr_time) {
      return (AODVC_EXPTIME_UPDATE_SHORT);
    }
    else if (exp_time == OPC_DBL_INFINITY) {
      return (AODVC_EXPTIME_UPDATE_INFINITE);
    }

    return (AODVC_EXPTIME_UPDATE_FAILURE);
  }

  return (AODVC_EXPTIME_UPDATE_SUCCESS);
}
```

B.2.8 aodv_route_table_pre_route_expiry_timer_update Function

The aodv_route_table_pre_route_expiry_timer_update() function is called by the functions, aodv_rte_rreq_pkt_arrival_handle() and the aodv_rte_route_table_entry_update(). This function is called by the intermediate nodes to update the pre-route expiry timer for the received RREQ, so that the node will disable the “pre-route state” of the route entry to the source. This is needed for the node to determine what to do to the received RREQ, RREP or data packets, as different updates are needed depending on whether the route is in the “pre-route state”.

Code B.2.8: aodv_route_table_pre_route_expiry_timer_update() Function

```
void aodv_route_table_pre_route_expiry_timer_update (
                    route_entry, dest_addr, exp_time, code) {

  cancel existing event(s) in route_entry->pre_route_expiry_timer_evhandle;

  if (route_entry->pre_route_state is TRUE &&
      curr_time <= exp_time && exp_time < OPC_DBL_INFINITY) {
    /* If pre-route state is activated, and the exp_time is between */
    /* now and infinity, set an interrupt to apply pre-route        */
    /* information when the pre_route_timer expires.                */
    call op_intrpt_schedule_call (exp_time, code,
        aodv_rte_entry_pre_route_timer_handle, route_entry) to schedule an
        interrupt to apply pre-route information to the route_entry, and store
        the event handler in route_entry->pre_route_expiry_timer_evhandle.
    }
  else {
    /* Do not schedule any event if the exp_time is */
    /* before the current time or if it is infinity.*/
    if (route_entry->pre_route_state if FALSE) {
      if (exp_time < now) {
        return (AODVC_EXPTIME_UPDATE_SHORT_NOPRERTE);
      }
      else if (exp_time == OPC_DBL_INFINITY) {
        return (AODVC_EXPTIME_UPDATE_INFINITE_NOPRERTE);
      }
      else {
        return (AODVC_EXPTIME_UPDATE_NOPRERTE);
      }
    }
    else {
      if (exp_time < now) {
        return (AODVC_EXPTIME_UPDATE_SHORT);
      }
      else if (exp_time == OPC_DBL_INFINITY) {
        return (AODVC_EXPTIME_UPDATE_INFINITE);
      }
    }

    return (AODVC_EXPTIME_UPDATE_FAILURE);
    }

  return (AODVC_EXPTIME_UPDATE_SUCCESS);
}
```

B.3 Send RREP

When an interrupt to send a RREP is called, the function, aodv_rte_rrep_send_timer_handle() in Code B.3, is executed. This function calls aodv_route_table_apply_pre_route() (Code B.3.1) to update the route entry, and then send a RREP back to the source of the RREQ, which is the destination of the route entry.

Code B.3: aodv_rte_rrep_send_timer_handle() Function

```
void aodv_rte_rrep_send_timer_handle (route_entry , code) {

  call ip_cmn_rte_table_dest_prefix_addr_get (route_entry->dest_prefix) to
      get the dest_addr;
  call aodv_route_table_entry_get() to get the src_route_entry;
  call aodv_route_table_entry_get() to get the dest_route_entry;

  if (code is AODVC_PRE_RREP_SEND_ACTIVE &&
          src_route_entry exists &&
          dest_route_entry exists) {
    /* If the intermediate node has a route to the   */
    /* destination and destination only flag is off, */
    /* then, apply pre-route info and send a RREP.   */
    call aodv_route_table_apply_pre_route() to apply pre-route information
        for the src_route_entry;
    call aodv_route_table_apply_pre_route() to apply pre-route information
        for the dest_route_entry;
    call aodv_rte_route_reply_send() to send RREP to the source;
  }
  else if (src_route_entry exists) {
    /* If this is the destination of the RREQ, then */
    /* apply pre-route info and send a RREP.        */
    call aodv_route_table_apply_pre_route() to apply pre-route information
        for the src_route_entry;
    call aodv_rte_route_reply_send() to send RREP to the source;
  }
}
```

B.3.1 aodv_route_table_apply_pre_route Function

Code B.3.1: aodv_route_table_apply_pre_route() Function

```
void aodv_route_table_apply_pre_route (
                    route_table , route_entry , dest_addr , stage) {
  // Cancel route_expiry_update_evhandle
  route_entry_ptr->route_expiry_update_timer = 0;
  op_ev_cancel_if_pending (route_entry_ptr->route_expiry_update_evhandle);

  if (route_entry->route_entry_state is VALID ||
      (route_entry_state is INVALID &&
            route_entry->pre_route_route_expiry_time > curr_time)) {
    /* Apply pre-route info to the current route info as follow. */

    // Cancel the following future events
    op_ev_cancel_if_pending (route_entry->route_expiry_evhandle);
    op_ev_cancel_if_pending (route_entry->pre_route_start_timer_evhandle);
    op_ev_cancel_if_pending (route_entry->pre_route_expiry_timer_evhandle);

    if (route_entry is in the IP routing table) {
      call Inet_Cmn_Rte_Table_Entry_Update() to update the IP routing table
          with pre-route routing information.
    }
    else {
      call Inet_Cmn_Rte_Table_Entry_Add_Options() to add routing information
          into the IP routing table.
    }

    /* Free the current next hop address if needed */
    remove route_entry->next_hop_addr;

    /* Set the following route_entry variables */
    route_entry->route_entry_state = VALID;
    route_entry->dest_seq_num = route_entry->pre_route_dest_seq_num;
    route_entry->valid_dest_sequence_number_flag =
                route_entry->pre_route_valid_dest_sequence_number_flag;
    route_entry->next_hop_addr = copy(route_entry->pre_route_next_hop_addr);
    route_entry->next_hop_port_info =
                route_entry->pre_route_next_hop_port_info;
    route_entry->hop_count = route_entry->pre_route_hop_count;
    route_entry->route_expiry_time =
                route_entry->pre_route_route_expiry_time;
    route_entry->new_route_expiry_time =
                route_entry->pre_route_route_expiry_time;
    route_entry->link_expiry_time = route_entry->pre_route_link_expiry_time;
    route_entry->new_link_expiry_time =
                route_entry->pre_route_link_expiry_time;
    if ((route_entry->route_expiry_time -
                      route_entry->link_expiry_time < 0.001)||
        (route_entry->route_expiry_time -
                      route_entry->link_expiry_time > 0.001))
      route_entry->is_shortest_link_duration = TRUE;
    route_entry->route_request_time = curr_time;
    if (route_entry->route_begins_time == -1 ||
        (route_entry_state is INVALID &&
            route_entry->pre_route_route_expiry_time > curr_time))
      route_entry->route_begins_time = curr_time;
    route_entry->pre_route_start_timer = 0;
    route_entry->route_expiry_update_timer = 0;

    /* Free the pre-route next hop address if needed */
    remove route_entry->pre_route_next_hop_addr;

    /* Reset the following pre-route information */
    if (route_entry->route_entry_state is VALID) {
      route_entry->pre_route_state = OPC_FALSE;
      route_entry->pre_route_ready = OPC_FALSE;
    }

    if (route_entry->route_expiry_time > curr_time &&
        route_entry->route_expiry_time < INFINITY) {
      if ((route_entry->is_source_node is TRUE) &&
          (curr_time - route_entry->last_app_pkt_to_dest_time <
                        route_entry->last_pkt_to_dest_period)) {
        /* If this is the sourcee, set the pre_route_start_timer */
        call aodv_route_table_pre_route_start_timer_update() to set
            the pre_route_start_timer;
      }

      /* Schedule an interrupt at the route expiry time. */
      call op_intrpt_schedule_call() to schedule an interrupt at
          route_entry->route_expiry_time;
    }
  }
}
```

B.4 Process RREP

Upon receiving a RREP packet from the destination, the aodv_rte_rrep_pkt_arrival_handle() function is called to process the received RREP packet. This involve creating or updating the route entries for the destination and the previous hop of the RREP packet, and then forward the RREP packet to the next hop along the path back to the source.

Code B.4: aodv_rte_rrep_pkt_arrival_handle() Function

```
void aodv_rte_rrep_pkt_arrival_handle (..., AodvT_Packet_Option* tlv_options_ptr) {
  set rrep = (AodvT_Rrep*) tlv_options_ptr->value_ptr;
  ...
  /* Added for Prediction */
  if (predictive_aodv) {
    call predict_linkchange_time (prev_hop) to predict linkchange_time;
    if (linkchange_time < curr_time) {
      linkchange_time = curr_time;
    }
    pathchange_time = min (rrep->pathchange_time, linkchange_time);
    rrep->pathchange_time = pathchange_time;

    if (linkchange_time < rrep->lifetime)
      rrep->lifetime = linkchange_time - curr_time;
  }
  /* Prediction Ends */

  call aodv_rte_neighbor_connectivity_table_update (prev_hop, FALSE) to
      add or update the neighbour connectivity table;

  call aodv_rte_route_table_entry_update (..., RECV_RREP) to update
      the route_entry to prev_hop;

  /* Added for Prediction */
  if (predictive_aodv &&
      pathchange_time <= curr_time) {
    /* If route request's pathchange time is too short, discard the RREP. */
    discard RREP Pkt;
    return;
  }
  /* Prediction Ends */

  rrep->hop_count++;
  if (rrep->hop_count > net_diameter) {
    /* If hop count is too large, destroy the route request. */
    discard RREP Pkt;
    return;
  }

  ...

  call aodv_route_table_entry_get (route_table, rrep->dest_addr) to
      get destination route_entry;

  if (route_entry to the destination does not Exist) {
    /* Modified for Prediction, as the aodv_route_table_entry_create() */
    /* function has been modified.                                     */
    call aodv_route_table_entry_create() to create a
        route_entry for rrep->dest_addr;
  set route_updated = TRUE;
  /* Modification Ends */
  call aodv_rte_all_pkts_to_dest_send() to send packets
      to rrep->dest_addr;
}
else {
  set dest_seq_num = route_entry->dest_seq_num;
  set hop_count = route_entry->hop_count;
  set route_entry_state = route_entry->route_entry_state;

  /* Added for Prediction */
  if (predictive_aodv)
    pre_route_state = route_entry->pre_route_state;
  /* Prediction Ends */

  /* Modified for Prediction */
  if ((rrep->dest_seq_num > dest_seq_num) ||
      ((rrep->dest_seq_num == dest_seq_num) &&
        (route_entry_state is INVALID)) ||
      (predictive_aodv && (rrep->dest_seq_num == dest_seq_num) &&
        (rrep->pathchange_time > route_entry->route_expiry_time + 0.1)) ||
      (predictive_aodv && (rrep->dest_seq_num == dest_seq_num) &&
        (rrep->pathchange_time >
                    route_entry->pre_route_route_expiry_time + 0.1)) ||
      (standard_aodv && (rrep->dest_seq_num == dest_seq_num) &&
        (rrep->hop_count < route_entry->hop_count))) {
    /* Record last route duration for both AODV & AODV-PP. */
    if (predictive_aodv) {
      if (route_entry->next_hop_addr is not prev_hop_addr ||
          route_entry->hop_count != rrep_option_ptr->hop_count ||
          (track_info_ptr->predict_method is PERFECT_PREDICTION &&
            route_entry->route_expiry_time != rrep->pathchange_time)) {
        /* If route has changed, do the following. */
        if (rrep->route_entry_state is VALID &&
            route_entry->last_app_pkt_to_dest_time > 0 &&
            (route_entry->route_begins_time <
                          route_entry->last_app_pkt_to_dest_time ||
              curr_time - route_entry->last_app_pkt_to_dest_time <
                          last_pkt_to_dest_period) &&
            route_entry->route_begins_time > 0) {
          /* If route has been used over the life of the last route, */
          /* record last route duration statistics.                  */
        }
        set route_entry->route_begins_time = curr_time;
      }
    }
    else {
      if (route_entry->next_hop_addr is not prev_hop_addr ||
          route_entry->hop_count != rrep_option_ptr->hop_count) {
        /* If route has changed, do the following. */
      if (route_entry_state is VALID &&
          route_entry->last_app_pkt_to_dest_time > 0 &&
          (route_entry->route_begins_time <
                       route_entry->last_app_pkt_to_dest_time ||
            curr_time - route_entry->last_app_pkt_to_dest_time <
                       last_pkt_to_dest_period) &&
          route_entry->route_begins_time > 0) {
        /* If route has been used over the life of the last route, */
        /* record last route duration statistics.                   */
      }
      set route_entry->route_begins_time = curr_time;
    }
  }

  if (standard_aodv ||
      route_entry_state is not VALID ||
      rrep->dest_seq_num > dest_seq_num ||
      (predictive_aodv && (rrep->dest_seq_num == dest_seq_num) &&
            (rrep->pathchange_time > route_entry->route_expiry_time))) {

    set route_entry->hop_count = rrep_option_ptr->hop_count;
    set route_entry->next_hop_port_info = in_port_info;
    set route_entry->new_link_expiry_time = linkchange_time; // for AODV-PU

    if (route_entry_state is not VALID ||
        (predictive_aodv && !pre_route_state)) {
      /* If it is not a valid route or it is not in a pre-route state, */
      /* update the pre-route variables in the AODV model.             */
      set route_entry->pre_route_hop_count = rrep->hop_count;
      set route_entry->pre_route_next_hop_port_info = in_port_info;
      set route_entry->pre_route_route_expiry_time = rrep->pathchange_time;
      set route_entry->pre_route_link_expiry_time = linkchange_time;
    }

    if (predictive_aodv && pre_route_state && route_entry_state is VALID) {
      call Inet_Cmn_Rte_Table_Entry_Update() to update the IP routing table;
      free route_entry->next_hop_addr;
      set route_entry->dest_seq_num = rrep->dest_seq_num;
      set route_entry->next_hop_addr = prev_hop_addr;
      set route_entry->route_request_time = curr_time;
      if (route_entry->route_begins_time == -1)
        set route_entry->route_begins_time = curr_time;
    }
    else {
      call aodv_route_table_entry_next_hop_update (route_table, route_entry,
          rrep->dest_seq_num, prev_hop_addr, rrep->hop_count, in_port_info)
          to update the next hop and the hop count of the route_entry with
          the new values for both standard and pre-route information;
    }

    if (route_entry_state is not VALID) {
      call aodv_route_table_entry_state_set (route_table, route_entry,
          rrep->dest_addr, VALID) to set the route to be a valid route;

      if (predictive_aodv)
        call aodv_route_table_apply_pre_route (...) to replace the route
            information of the route_entry with pre_route information;

      if (route_entry->is_source_node)
        add route setup statistics;
    }

    /* Mark the destination sequence as valid */
    aodv_route_table_entry_param_set (route_entry,
        AODVC_ROUTE_ENTRY_VALID_SEQ_NUM_FLAG, TRUE);

    /* Set the destination sequence number to that in the RREP message  */
    aodv_route_table_entry_param_set (route_entry,
        AODVC_ROUTE_ENTRY_DEST_SEQ_NUM, rrep->dest_seq_num);

    if (standard_aodv)
      call aodv_route_table_entry_route_expiry_time_update (...,
          rrep->lifetime, AODVC_ROUTE_ENTRY_INVALID) to set the
          route expiry time;
    els if (predictive_aodv)
      call aodv_route_table_entry_route_expiry_time_update (...,
          rrep->pathchange_time - curr_time, AODVC_ROUTE_ENTRY_INVALID)
          to set the route expiry time;
  }

  if (predictive_aodv && !pre_route_state) {
    if (rrep->pathchange_time < route_entry->route_expiry_time) {
      rrep->pathchange_time = route_entry->route_expiry_time;
    }

    /* Check if the destination is a neighbour. */
    neighbor_info = (AodvT_Conn_Info *) inet_addr_hash_table_item_get (
        neighbor_connectivity_table, &rrep->dest_addr);

    if (neighbor_info exists) {
      /* If destination is the neighbour of this node, remove the neighbour. */
      op_ev_cancel_if_pending (neighbor_info->conn_expiry_handle);

      call inet_addr_hash_table_item_remove (neighbor_connectivity_table,
          &(neighbor_info->neighbor_address)) to remove the entry for this
          node from the neighbor_connectivity_table;
      free (neighbor_info);
      }

    call aodv_route_table_apply_pre_route (route_table, route_entry,
                rrep->dest_addr, RECV_RREP) to apply pre-route information;
          }

          call aodv_rte_all_pkts_to_dest_send () to send all queued packets to
            the destination;

          set route_updated = TRUE;
        }
        /* Modification Ends */
      }

      if (reached_src) {
        ...
      }

      get fwd_route_entry = aodv_route_table_entry_get (route_table, rrep->src_addr);
      if (fwd_route_entry Exists &&
                (fwd_route_entry->route_entry_state is not INVALID)) {
        /* If the forward route entry back to the source exists and is valid, */
        /* forward the route reply.                                           */

        set next_hop_addr = fwd_route_entry->next_hop_addr;
        set hop_count = fwd_route_entry->hop_count;
        set route_expiry_time = fwd_route_entry->route_expiry_time;

        /* Update the reverse route expiry time before sending out the RREP */
        set existing_lifetime = route_expiry_time - curr_time;

        /* active_route_timeout = my_route_timeout/2, use this according to RFC */
        set lifetime = aodv_rte_max_find (existing_lifetime, my_route_timeout/2);

        /* Added for Prediction */
        if (predictive_aodv) {
          set lifetime = fwd_route_entry->route_expiry_time - curr_time;

          /* Print previous route duration */
          if ((fwd_route_entry->next_hop_addr !=
                                fwd_route_entry->pre_route_next_hop_addr) ||
              (fwd_route_entry->hop_count != fwd_route_entry->pre_route_hop_count) ||
              (track_info_ptr->predict_method is PERFECT_PREDICTION &&
                fwd_route_entry->route_expiry_time !=
                                  fwd_route_entry->pre_route_route_expiry_time)) {
            if (fwd_route_entry->route_entry_state is VALID &&
                fwd_route_entry->last_app_pkt_to_dest_time > 0 &&
                (fwd_route_entry->route_begins_time <
                                  fwd_route_entry->last_app_pkt_to_dest_time ||
                  curr_time - fwd_route_entry->last_app_pkt_to_dest_time <
                                  last_pkt_to_dest_period) &&
                  fwd_route_entry->route_begins_time > 0) {
              /* If route has been used over the life of the last route, */
          /* record last route duration statistics here.          */
        }

        /* set route_begins_time every time route update is to be accomplished */
        set fwd_route_entry->route_begins_time = curr_time;
      }

      set next_hop_addr = fwd_route_entry->pre_route_next_hop_addr;
      set hop_count = fwd_route_entry->pre_route_hop_count;
      set route_expiry_time = fwd_route_entry->pre_route_route_expiry_time;
      set lifetime = fwd_route_entry->pre_route_route_expiry_time - curr_time;

      /* Mark the destination sequence as valid */
      aodv_route_table_entry_param_set (fwd_route_entry,
          AODVC_ROUTE_ENTRY_VALID_SEQ_NUM_FLAG, TRUE);

      set fwd_route_entry->pre_route_state = FALSE;

      /* Apply pre-route info */
      aodv_route_table_apply_pre_route (route_table, fwd_route_entry,
          rrep->src_addr, PRER_RREP);

      call aodv_rte_all_pkts_to_dest_send (...) to send all packets queued to
          the destination of the fwd_route_entry;

      /* Check if the destination is a neighbour. */
      neighbor_info = (AodvT_Conn_Info *) inet_addr_hash_table_item_get (
          neighbor_connectivity_table, &rrep->dest_addr);

      if (neighbor_info exists) {
        /* If destination is the neighbour of this node, remove the neighbour. */
        op_ev_cancel_if_pending (neighbor_info_ptr->conn_expiry_handle);

        call inet_addr_hash_table_item_remove (neighbor_connectivity_table,
            &(neighbor_info->neighbor_address)) to remove the entry for this
            node from the neighbor_connectivity_table;
        free (neighbor_info);
        }

      /* Indicate that the route has been created */
      set route_updated = TRUE;
    }
    /* Prediction Ends */
    else {
      /* If standard AODV, update route expiry time. */
      call aodv_route_table_entry_route_expiry_time_update (fwd_route_entry,
          rrep->src_addr, lifetime, AODVC_ROUTE_ENTRY_INVALID) to update the
          route expiry time of the forward route entry;
    }

```

```
      ...

      record statistics for forwarding RREP;
    }

    ...
}
```

B.5 Start and End PreRREQ

The aodv_rte_entry_pre_route_timer_handle() function is called via an interrupt in two occasions: one is when it is time to start sending a PreRREQ; and the other one is when PreRREQ has expired. In the first scenario this function creates a RREQ packet and broadcasts it to search for a route before the route expires. The second scenario involves route entry update.

Code B.5: aodv_rte_entry_pre_route_timer_handle() Function

```
void aodv_rte_entry_pre_route_timer_handle (route_entry, code) {
  if (route_entry Exists) {
    if (code == AODVC_PRE_RREQ_START &&
        route_entry->is_source_node is TRUE) {
      /* First scenario - This is the source node, send PreRREQ */

      initialise linkchange_time = -1;
      initialise pathchange_time = INFINITY;

      set ttl = (int)min((route_entry->hop_count + (2*ttl_increment)), net_diameter);
      set request_expiry_time = 2.0 * node_traversal_time * (ttl + timeout_buffer);

      if (curr_time - route_entry->last_app_pkt_to_dest_time <
                              use_duration_threshold &&
          route_entry->route_expiry_time > curr_time) {
        /* If the route is still in use recently, and if this has been */
        /* the source, update the pre_route_expiry_timer and the       */
        /* pre_route_state to TRUE.                                    */
        set route_entry->pre_route_expiry_timer = curr_time + request_expiry_time;
        set route_entry->pre_route_state = TRUE;
      }
      else if (route_entry_ptr->route_expiry_time < curr_time){
        calculate and record route statistics;
      }

      /* Send a RREQ packet for Pre-Route */
      if (RREQ is not in the request_table) {
        call aodv_rte_route_request_send (route_entry, dest_addr, ttl,
            request_expiry_time, 0) to send RREQ;
      }

      if (route_entry->pre_route_precursor_lptr is not empty) {
        remove all precursor nodes from the list;
      }
    }
    else if (code is AODVC_PRE_RREQ_EXPIRY) {
      /* Second scenario - Pre-Route RREQ has expired. */
      /* Update the route entry information.          */

      set route_entry->pre_route_state to FALSE;


      if (route_entry->next_hop_addr == route_entry->pre_route_next_hop_addr) &&
          route_entry->hop_count == route_entry->pre_route_hop_count) {
        /* If the pre-route info for the next hop are the same as the current */
        /* route info, copy the pre-route info to the route entry.            */
        set route_entry_ptr->dest_seq_num =
                              route_entry_ptr->pre_route_dest_seq_num;
        set route_entry_ptr->new_route_expiry_time =
                              route_entry_ptr->pre_route_route_expiry_time;
        }
    }
  }
}
```

B.6 Update and Verification

Although the expiry time is meant to be predicted accurately, an update is used, upon receiving of a data packet, to verify whether the predicted link change time is correct.

Code B.6: aodv_rte_data_routes_expiry_time_update() Function

```
void aodv_rte_data_routes_expiry_time_update (Packet* pkptr) {
  ...
  call aodv_route_table_entry_get() to get src_route_entry;
  call aodv_route_table_entry_get() to get dest_route_entry;
```

```

if (src_route_entry exists) {
  ...
  if (standard_aodv) {
    update route_entry(s) as shown in code A.2;
  }
  else {
    /* If prediction is used, update src_route_entry as follow. */

    /* set linkchange_time */
    call predict_linkchange_time(prev_hop_addr) to predict linkchange_time;

    if (src_route_entry->pre_route_state is TRUE) {
      if (src_route_entry->pre_route_expiry_timer > curr_time) {
        call aodv_route_table_entry_pre_route_timer_update() to ensure
            the pre_route information of the route_entry will be applied
            when the src_route_entry->pre_route_expiry_timer expires;
      else {
        set src_route_entry->pre_route_state to FALSE;
        call aodv_route_table_apply_pre_route() to replace the route
            information of the route_entry with pre_route information;
      }
    }
    else {
      if (src_route_entry->next_hop_addr is prev_hop_addr) {
        /* set pathchange_time */
        set pathchange_time = src_route_entry->route_expiry_time;
        if (linkchange_time < pathchange_time)
          set pathchange_time = linkchange_time;
        set pathchange_timeout = pathchange_time - curr_time;
        if (pathchange_timeout <= 0)
          set pathchange_timeout = 0;

        /* Update the expiry time of the route entry. */
        if (src_route_entry->pre_route_state is FALSE)
          call aodv_route_table_entry_expiry_time_update() to update the
              src_route_entry->route_expiry_time to pathchange_timeout;
      }

      /* Update the entry for the previous hop's route entry. */
      call aodv_route_table_entry_get() to get the prev_hop_route_entry;
      if (prev_hop_route_entry is not Empty &&
          prev_hop_route_entry->pre_route_state is FALSE &&
          src_addr is not prev_hop_addr &&
          prev_hop_route_entry->next_hop_addr is prev_hop_addr) {
        call aodv_route_table_entry_expiry_time_update (route_table,
            prev_hop_route_entry, prev_hop_addr, linkchange_timeout,
            AODVC_ROUTE_ENTRY_INVALID, RECV_PKT, aodv_type) to update
            the prev_hop_route_entry->route_expiry_time;
      }
    }
  }
}
else if (this node is not the source of the packet) {
  /* If src_route_entry does not exist and if */
  /* this is not the source node, a new route */
  /* entry needs to be created.               */
  get the port and subnet mask information;

  /* The following code is added to set the pathchange_timeout   */
  /* that will be used in creating a route_entry for the source. */
  if (standard_aodv) {
    pathchange_timeout = route_table_ptr->route_expiry_time;
  }
  else {
    /* set linkchange_time */
    call predict_linkchange_time(prev_hop_addr) to predict linkchange_time;

    /* set pathchange_time */
    set pathchange_time = the last predicted pathchange_time;
    if (linkchange_time < pathchange_time)
      set pathchange_time = linkchange_time;
    set pathchange_timeout = pathchange_time - curr_time;
    if (pathchange_timeout <= 0)
      set pathchange_timeout = 0;
  }

  call aodv_route_table_entry_create() to create a new
      route_entry for the source;
}

if (dest_route_entry exists) {
  ...
  if (standard_aodv) {
    update route_entry(s) as normal;

    if (src_route_entry does not exist &&
            this node is the source of the packet) {
      set dest_route_entry->is_source_node to TRUE;
      set dest_route_entry->last_app_pkt_to_dest_time = curr_time;
    }
  }
  else {
    /* If prediction is used, update dest_route_entry as follow. */

    /* set linkchange_time */
    call predict_linkchange_time(next_hop_addr) to predict linkchange_time;

    if (dest_route_entry->pre_route_state is TRUE) {
      if (dest_route_entry->pre_route_expiry_timer > curr_time) {
        call aodv_route_table_entry_pre_route_timer_update() to ensure
            the pre_route information of the route_entry will be applied
            when the dest_route_entry->pre_route_expiry_timer expires;
      else {
        set dest_route_entry->pre_route_state to FALSE;
        call aodv_route_table_apply_pre_route() to replace the route
            information of the route_entry with pre_route information;
      }
    }
    else {
      if (dest_route_entry->next_hop_addr is next_hop_addr) {
        /* set pathchange_time */
        set pathchange_time = dest_route_entry->route_expiry_time;
        if (linkchange_time < pathchange_time)
          set pathchange_time = linkchange_time;
        set pathchange_timeout = pathchange_time - curr_time;
        if (pathchange_timeout <= 0)
          set pathchange_timeout = 0;

        /* Update the expiry time of the route entry. */
        if (dest_route_entry->pre_route_state is FALSE)
          call aodv_route_table_entry_expiry_time_update (route_table,
              dest_route_entry, dest_addr, pathchange_timeout,
              AODVC_ROUTE_ENTRY_INVALID, RECV_PKT, aodv_type) to update
              the dest_route_entry->route_expiry_time;
      }

      /* Update the entry for the next hop's route entry. */
      call aodv_route_table_entry_get() to get the next_hop_route_entry;
      if (next_hop_route_entry is not Empty &&
          next_hop_route_entry->pre_route_state is FALSE &&
          dest_addr is not next_hop_addr &&
          next_hop_route_entry->next_hop_addr is next_hop_addr) {
        call aodv_route_table_entry_expiry_time_update (route_table,
            next_hop_route_entry, next_hop_addr, linkchange_timeout,
            AODVC_ROUTE_ENTRY_INVALID, RECV_PKT, aodv_type) to update
            the next_hop_route_entry->route_expiry_time;
      }
    }

    if (src_route_entry does not exist &&
          this node is the source of the packet) {
      if (dest_route_entry->pre_route_state is FALSE &&
          curr_time < dest_route_entry->route_expiry_time -
                                  MAX_RING_TRAVERSAL_TIME &&
          (dest_route_entry->is_source_node is FALSE ||
            dest_route_entry_->pre_route_start_timer !=
                dest_route_entry->route_expiry_time -
                                  MAX_RING_TRAVERSAL_TIME)) {
        call aodv_route_table_entry_pre_route_timer_update() to
```

```
              re-schedule the pre_route_start_timer interrupt to
              dest_route_entry->route_expiry_time - MAX_RING_TRAVERSAL_TIME;
        }

        set dest_route_entry->is_source_node to TRUE;
        set dest_route_entry->last_app_pkt_to_dest_time = curr_time;
      }
    }
  }

  update precursor lists as shown in code A.2;
}
```

B.7 Pre-Route RREQ Start Timer

The aodv_route_table_pre_route_start_timer_update() function is called by any source node to update the pre-route start time and to trigger an interrupt to start sending RREQ before the route expires. This function is only needed when the nodes have data packets to send to the destination. Therefore this function can be called by a number of functions such as aodv_rte_rrep_send_timer_handle(), aodv_rte_rrep_pkt_arrival_handle(), aodv_rte_data_routes_expiry_time_update() and aodv_rte_all_pkts_to_dest_send().

Code B.6: aodv_route_table_pre_route_start_timer_update() Function

```
void aodv_route_table_pre_route_start_timer_update (route_table ,
                    route_entry , dest_addr , rte_exp_time , code) {
  set pre_route_start_time = rte_exp_time - (2.0 *
      route_table->node_traversal_time *
      (route_table->net_diameter + route_table->timeout_buffer));

  cancel existing event(s) in route_entry->pre_route_start_timer_evhandle;

  if (curr_time <= pre_route_start_time &&
      pre_route_start_time < OPC_DBL_INFINITY) {
    /* If pre-route state is activated , and the exp_time is between */
    /* now and infinity , set an interrupt to send pre-route RREQ    */
    /* at pre_route_start_time.                                       */
    set route_entry->pre_route_start_timer = pre_route_start_time;
    call op_intrpt_schedule_call() to schedule an interrupt call to send
```

```
        a PreRREQ before the route expires at pre_route_start_time;
  }
  else {
    /* Do not schedule any event if the exp_time is */
    /* before the current time or if it is infinity.*/
    if (pre_route_start_time < now) {
      return (AODVC_EXPTIME_UPDATE_SHORT);
    }
    else if (exp_time == OPC_DBL_INFINITY) {
      return (AODVC_EXPTIME_UPDATE_INFINITE);
    }

    return (AODVC_EXPTIME_UPDATE_FAILURE);
  }

  return (AODVC_EXPTIME_UPDATE_SUCCESS);
}
```

Appendix C

Modifications to AODV-PP for AODV-PU

The modifications that have been made to AODV-PP to develop AODV-PU are included in this appendix. These modifications are the major modifications made to the model, they are not all the modifications made.

C.1 Packet Structures

The RREQ, RREP, RERR and HELLO packets used for AODV-PU are the same as AODV-PP. However, AODV-PU requires to send route prediction updates, a new packet type, RUPDATE, is added.

C.1.1 RUPDATE Packet Structure

The 96-bit RUPDATE packet structure illustrated in Table C.1 consists of the following fields:

1. **Packet Type**: 6;
2. **Destination Address**: The IP address of the destination to be updated with a new destination pathchange time; and
3. **Destination Pathchange Time**: The new pathchange time of the destination in the route entry.

00	1	2	3	4	5	6	7	8	9	10	1	2	3	4	5	6	7	8	9	20	1	2	3	4	5	6	7	8	9	30	1
Packet Type								Reserved																							
Destination Address																															
Destination Pathchange Time																															

Table C.1: *RUPDATE Packet Structure.*

C.2 Route Expiry Time Update Timer

A link and route expiry prediction update is accomplished upon receiving a data packet. A new field, route_exp_time, is added to the IP datagram fields structure, IpT_Dgram_Fields (in the ip_dgram_sup.h file), in OPNET to store the route expiry time. This field is initiated as infinity (or the largest value), by the source node. When a data packet came from a neighbouring node, the duration of that link or the link expiry time will be estimated, and the duration of the route to the source will also be evaluated, by comparing the link expiry time with the route expiry time, and the route expiry time field in the IP packet is updated after each hop. The route expiry time of all the route entries that use the previous node as the next hop will also be compared with the estimated expiry time of this link. If the link expires before the route expiry time, the new_route_expiry_time field of the route_entry needs to be updated, and RUPDATE is then scheduled to be sent. If the received data packet is to be forwarded to the next node, the link duration of the forwarding link will also be estimated. This is done because when a packet is sent, it is assumed that cross-layer information can be used, and that Request-To-Send (RTS) and Clear-To-Send (CTS) frames are assumed to be exchanged in the Medium Access Control (MAC) Layer, and therefore, the link duration can be estimated from the signal strength information gathered through this and the previous RTS/CTS exchange upon sending the data packets. The changes made to do prediction update is illustrated in the pseudo code of the aodv_rte_data_routes_expiry_time_update() function in Code C.2 below.

Code C.2: aodv_rte_data_routes_expiry_time_update() Function

```
void aodv_rte_data_routes_expiry_time_update (...) {
  ...
  get a list of all the route entries, rentry_lptr;

  call aodv_route_table_entry_get() to get src_route_entry;
  call aodv_route_table_entry_get() to get dest_route_entry;

  if (src_route_entry exists) {
    get prev_hop_addr from the received packet;
    ...
    if (standard_aodv) {
      update route_entry(s) as shown in code A.2;
    }
    else {
      /* If prediction is used, update src_route_entry as follow. */

      /* predict the linkchange_time by calling predict_linkchange_time() */
      set linkchange_time = predict_linkchange_time(prev_hop_addr);

      /* Update ip_field->route_exp_time (A field added to IP datagram) */
      if (ip_field->route_exp_time > linkchange_time)
        ip_field->route_exp_time = linkchange_time;

      /* set pathchange_time */
      set pathchange_time = ip_field->route_exp_time;
      set pathchange_timeout = pathchange_time - curr_time;
      if (pathchange_timeout <= 0)
        set pathchange_timeout = 0;

      /* Update all the route entries that use prev_hop_addr */
      /* as their next hop to the destination.              */
      for (rte_entry_count=0; rte_entry_count<num_routes; rte_entry_count++) {
        get rte_entry = op_prg_list_access (rentry_lptr, rte_entry_count);
        get rte_entry_dest_addr = ip_cmn_rte_table_dest_prefix_addr_get
                                          (rte_entry->dest_prefix);

        if (rte_entry_dest_addr is not dest_addr &&
            rte_entry_dest_addr is not next_hop_addr &&
            rte_entry->next_hop_addr is prev_hop_addr) {
          call aodv_rte_expiry_time_update_process() to decide whether
              RUPDATE or RERR needs to be sent to the precursor nodes,
              if so, set an interrupt;
        }
      }
    }
  }
  else if (this node is not the source of the packet) {
    /* If src_route_entry does not exist and if this is not the source */
    /* node, a new route entry needs to be created as follow.          */

    get the port and subnet mask information;

    /* The following code is added to set the pathchange_timeout   */
    /* that will be used in creating a route_entry for the source. */
    if (standard_aodv) {
      pathchange_timeout = route_table_ptr->route_expiry_time;
    }
    else {
      /* set linkchange_time */
      call predict_linkchange_time(prev_hop_addr) to predict linkchange_time;

      // Update the estimated route expiry time of the incoming data packet
      if (ip_field->route_exp_time > linkchange_time)
          ip_field->route_exp_time = linkchange_time;

      /* set pathchange_time */
      set pathchange_time = ip_field->route_exp_time;
      set pathchange_timeout = pathchange_time - curr_time;
      if (pathchange_timeout <= 0)
        set pathchange_timeout = 0;
    }

    call aodv_route_table_entry_create() to create a new route_entry
        for the source;
  }

  if (dest_route_entry exists) {
    get next_hop_addr from dest_route_entry->next_hop_addr;
    ...
    if (standard_aodv) {
      update route_entry(s) as in code B.6;
    }
    else {
      /* If prediction is used, update dest_route_entry as follow. */

      /* set linkchange_time */
      call predict_linkchange_time(next_hop_addr) to predict linkchange_time;

```

```
        /* set pathchange_time */
        set pathchange_time = min(dest_route_entry->route_expiry_time, linkchange_time);

        /* Update all the route entries that use next_hop_addr */
        /* as their next hop to the destination.              */
        for (rte_entry_count=0; rte_entry_count<num_routes; rte_entry_count++) {
          get rte_entry = op_prg_list_access (rentry_lptr, rte_entry_count);
          get rte_entry_dest_addr = ip_cmn_rte_table_dest_prefix_addr_get
                                            (rte_entry->dest_prefix);

          if (rte_entry_dest_addr is not src_addr &&
              rte_entry_dest_addr is not prev_hop_addr &&
              rte_entry->next_hop_addr is next_hop_addr) {
            call aodv_rte_expiry_time_update_process() to determine if a RUPDATE
                or RERR needs to be sent to precursors, if so, set an interrupt;
          }
        }

        if (src_route_entry does not exist &&
                this node is the source of the packet) {
          /* If the node is the sender of the packet (i.e. the packet arrives    */
          /* from the application layer, then the pre-route start timer needs    */
          /* to be updated according to the dest_route_entry->route_expiry_time. */
          if (dest_route_entry->pre_route_state is FALSE &&
              curr_time < dest_route_entry->route_expiry_time -
                                        MAX_RING_TRAVERSAL_TIME &&
              (dest_route_entry->is_source_node is FALSE ||
                dest_route_entry_->pre_route_start_timer !=
                    dest_route_entry->route_expiry_time -
                                        MAX_RING_TRAVERSAL_TIME)) {
            call aodv_route_table_entry_pre_route_timer_update() to re-schedule
                the pre_route_start_timer interrupt to
                dest_route_entry->route_expiry_time - MAX_RING_TRAVERSAL_TIME;
          }

          set dest_route_entry->is_source_node to TRUE;
          set dest_route_entry->last_app_pkt_to_dest_time = curr_time;
          set dgram_fields_ptr->route_exp_time = infinity;
        }
      }
    }

    update precursor lists as shown in code A.2;
}
```

Other functions that have been modified for AODV-PU to update the route expiry time include:

1. predict_linkchange_time() (lines 19 to 24 in Code B.2.1) - is used to

predict the link change time;

2. aodv_rte_expiry_time_update_process() (Code C.2.1) - is called to update the newly predicted route expiry time and schedule an interrupt to send a RUPDATE packet when required, or send a RERR if the route has already expired.
3. aodv_rte_route_error_process() (Code C.2.2) - is used to create and send a RERR packet;
4. aodv_rte_route_expiry_timer_update_handle() (Code C.2.3) - this is executed upon an interrupt call to update the route expiry to the new route expiry time, and to send a RUPDATE packet.

C.2.1 aodv_rte_expiry_time_update_process Function

This function is called to set an interrupt to update the route expiry time, and to send a RUPDATE for a route_entry if the predicted route expiry time has changed, or send a RERR if the route has already expired.

Code C.2.1: aodv_rte_expiry_time_update_process() Function

```
void aodv_rte_expiry_time_update_process (rte_entry, rte_entry_dest_addr,
                          next_hop_addr, is_predicted_pathchange_time_real,
                          is_link_broken, is_src) {
  /* Initialise some parameters */
  set is_rupdate_required = FALSE;
  set rte_entry->new_link_expiry_time = linkchange_time;

  /* Do not do anything if the route entry is not valid */
  if (rte_entry->route_entry_state != AodvC_Valid_Route) {
    return;
  }

  if (rte_entry->next_hop_addr is next_hop_addr) {
    if (predict_method is LET) {
      /* This part of the code is for inaccurate predictions. */

      /* Update the new_route_expiry_time in the route entry. */
      if (is_predicted_pathchange_time_real &&
          linkchange_time < rte_entry->route_expiry_time)
        rte_entry->new_route_expiry_time = linkchange_time;
      else if (is_predicted_pathchange_time_real)
        rte_entry->new_route_expiry_time = pathchange_time;
    else
      rte_entry->new_route_expiry_time = rte_entry->route_expiry_time;

    /* Reset the route_expiry_update_timer of the rte_entry. */
    rte_entry->route_expiry_update_timer = 0;
    call op_ev_cancel_if_pending () to cancel the pending interrupts
        associated with rte_entry->route_expiry_update_timer;

    /* Set the route_expiry_update_timer of the rte_entry, if the   */
    /* new_route_expiry_time is different to the route_expiry_time */
    /* more than the node_traversal_time (or any other constant).  */
    /* The route_expiry_update_timer is always set to be the min   */
    /* of the new and old times - (2*max_ring_traversal_time).     */
    if (rte_entry->new_route_expiry_time >
                 rte_entry->route_expiry_time + node_traversal_time ||
        rte_entry->new_route_expiry_time <
                 rte_entry->route_expiry_time - node_traversal_time)
      set rte_entry->route_expiry_update_timer =
          min(rte_entry->route_expiry_time, rte_entry->new_route_expiry_time) -
                                             (2 * max_ring_traversal_time);

    /* If the new_route_expiry_time is less than now, then set the */
    /* route_expiry_time of the route entry to the new expiry time,*/
    /* and set is_route_expired to TRUE. Otherwise, if route expiry*/
    /* time update is required, then if the precursor list is      */
    /* empty, update the route_expiry_time, or if there are        */
    /* precursors, set is_rupdate_required to TRUE to send RUPDATE.*/
    if (rte_entry->new_route_expiry_time < curr_time) {
      set rte_entry->route_expiry_time = rte_entry->new_route_expiry_time;
      set is_route_expired = OPC_TRUE;
    }
    else if (rte_entry->route_expiry_update_timer > 0) {
      if (rte_entry->precursor_lptr is empty)
        call aodv_route_table_entry_route_expiry_time_update() to
            set rte_entry->route_expiry_time = rte_entry->new_route_expiry_time;
      else
        set is_rupdate_required = TRUE;
    }
  }
  else {
    /* This part of the code is for perfect predictions. */

    /* Update the route_expiry_time of the route entry if needed. */
    if (is_predicted_pathchange_time_real is FALSE &&
        linkchange_time < rte_entry->route_expiry_time)
      set expiry_time = linkchange_time;
    else if (is_predicted_pathchange_time_real)
      set expiry_time = pathchange_time;

    call aodv_route_table_entry_route_expiry_time_update() to update the
```

```
        rte_entry->route_expiry_time = expiry_time;
    }
  }

  if (predict_method is LET &&
      rte_entry->pre_route_state is TRUE) {
    /* The following part of the code is for inaccurate predictions, */
    /* and if the pre_route_state is TRUE.                           */

    if (rte_entry->pre_route_next_hop_addr is the next_hop_addr) {
      /* If the next hop of the route entry is the same as the next_hop_addr, */
      /* update the pre_route_route_expiry_time of the route entry.           */
      if (is_predicted_pathchange_time_real is FALSE &&
          linkchange_time < rte_entry->pre_route_route_expiry_time)
        rte_entry->pre_route_route_expiry_time = linkchange_time;
    }
  }

  if (is_route_expired && rte_entry->route_expiry_time < curr_time)
    {
    /* If the route has expired, set the route expiry time to and */
    /* call an interrupt to set the route entry to INVALID, then  */
    /* sent a RERR, if required.                                  */

    Call aodv_route_table_entry_route_expiry_time_update() to set the
        rte_entry->route_expiry_time = curr_time, so the route will be set
        to INVALID immediate (at the next interrupt after this function);

    /* Send RERR if the route has been in use */
    if (rte_entry->last_pkt_to_dest_time > curr_time - 5) {
      if (is_link_broken && rte_entry_dest_addr == next_hop_addr)
        call aodv_rte_route_error_process() to send a RERR for
            AodvC_Link_Break_Detect;
      else if (!is_link_broken)
        call aodv_rte_route_error_process() to send a RERR for
              AodvC_Route_Broken;
      }

    /* Do not send RUPDATE *
    set is_rupdate_required = OPC_FALSE;
  }
  else if (is_rupdate_required) {
    /* If there is a need to send a RUPDATE, either send it now, or      */
    /* schedule to send it at a time set in the route_expiry_update_timer.*/

    if (rte_entry->route_expiry_update_timer < curr_time) {
      call aodv_rte_route_expiry_time_update_send (rte_entry) to send
          RUPDATE immediately;
    }
    else {
      call op_intrpt_schedule_call() to schedule an interrupt to send
          RUPDATE at rte_entry_ptr->route_expiry_update_timer;
    }
  }

  if (!is_route_expired && rte_entry->is_source_node &&
          rte_entry->next_hop_addr is next_hop_addr &&
          (rte_entry->pre_route_state is FALSE ||
           rte_entry->pre_route_start_timer > curr_time)) {
    /* If this is a source node, and the next hop address of the rte_entry */
    /* is the same as the one that is passing down to this function, reset */
    /* the pre_route_start_timer to either now or at a later time depending*/
    /* on the new_route_expiry_time recorded in the rte_entry.             */

    if (curr_time < rte_entry->new_route_expiry_time - max_ring_traversal_time) {
      call aodv_route_table_pre_route_start_timer_update() to set an
          interrupt to start searching for an alternative route immediately;
    }
    else if (curr_time < rte_entry->new_route_expiry_time) {
      rte_entry->pre_route_start_timer = curr_time;
      call op_ev_cancel_if_pending (rte_entry->pre_route_start_timer_evhandle)
          to cancel the previous pre-route start timer interrupt;
      call op_intrpt_schedule_call () to schedule an interrupt to start
          searching for a new route immediately;
    }
  }
}
```

C.2.2 aodv_rte_route_error_process Function

In standard AODV, the aodv_rte_route_error_process() function is called to process route error when there is a link break, a RERR packet has been received, or no route to the destination upon receiving a data packet. In AODV-PU, this function can be called upon disconnection of a route is detected. Hence, this case is added as indicated on line 4 of Code C.2.2.

Code C.2.2: aodv_rte_route_error_process() Function

```
1 void aodv_rte_route_error_process (...) {
2   ...
3   if (rerr_type == AodvC_Data_Packet_No_Route ||
4       rerr_type == AodvC_Route_Broken) {
5     ...
6   }
```

```
    ...
}
```

C.2.3 aodv_rte_route_expiry_timer_update_handle Function

This function is executed when an interrupt is triggered to send a RUPDATE.

Code C.2.3: aodv_rte_route_expiry_timer_update_handle() Function

```
void aodv_rte_route_expiry_timer_update_handle (
                            void* route_entry_vptr, int code) {
  set route_entry = (AodvT_Route_Entry *) route_entry_vptr;
  call aodv_rte_route_expiry_time_update_send (route_entry);
}
```

Besides the method mentioned above, the link duration can alternatively be estimated from the signal strength information gathered by receiving the periodically broadcast Management frames, such as the BEACON frames in WiFi. However, this is option is not implemented in AODV-PU, as it is not needed for the study.

C.3 Send RUPDATE

When it is time to send a RUPDATE packet, the aodv_rte_route_expiry_time_update_send() function (in Code C.3), will be called to construct and send RUPDATE.

Code C.3: aodv_rte_route_expiry_time_update_send() Function

```
void aodv_rte_route_expiry_time_update_send (route_entry) {
  initialise route_entry->route_expiry_update_timer = 0;

  if (predict_method is not LET ||
      route_entry->route_entry_state is not VALID ||
      (route_entry->last_route_expiry_update_time >
          curr_time - (2 * node_traversal_time *
          (route_entry->hop_count + timeout_buffer)))) {
    /* Update the route_expiry_update_timer and exit without sending  */
    /* RUPDATE, if inaccurate prediction is not used, or if the route */
    /* entry is not valid, or if the last_route_expiry_update_time    */
    /* happened not too long before the curr_time.                    */

    /* Update the route expiry update timer and re-schedule to send a */
    /* RUPDATE.                                                       */
    set route_entry->route_expiry_update_timer =
        route_entry->last_route_expiry_update_time + (2 * node_traversal_time *
        (route_entry->hop_count + timeout_buffer) + 0.1);

    call op_ev_cancel_if_pending() to cancel the pending interrupts
        associated with route_entry->route_expiry_update_timer;
    call op_intrpt_schedule_call() to re-schedule the interrupt to
        occur at route_entry->route_expiry_update_timer;

    return;
  }

  call ip_cmn_rte_table_dest_prefix_addr_get() to get the dest_addr from
      the route_entry;

  call aodv_route_table_entry_route_expiry_time_update() to update the
      route_entry->route_expiry_time = route_entry->new_route_expiry_time;

  if (route_entry->is_source_node) {
    if  (curr_time < route_entry->new_route_expiry_time -
                                    max_ring_traversal_time) {
      call aodv_route_table_pre_route_start_timer_update() to update the
          route_entry->pre_route_start_timer;
    }
    else {
      set route_entry->pre_route_start_timer = curr_time;
      cancel the existing route_entry->pre_route_start_timer interrupts;
      call op_intrpt_schedule_call () to schedule an interrupt to start
          searching for a new route immediately;
    }
  }

  if (No precursor exists) {
    /* Exit without sending RUPDATE */
    return;
  }

  /* Create the route update packet option  */
  set rupdate_option_ptr = aodv_pkt_support_rupdate_option_create
```

```
                            (route_entry->route_expiry_time, 1, dest_addr);

  /* Set the size for RUPDATE packet */
  set rupdate_size = (aodv_addressing_mode == InetC_Addr_Family_v4)? 96 : 192;

  call aodv_pkt_support_pkt_create(rupdate_option_ptr, rupdate_size) to set
      the route update option in the rupdate_pkptr packet;

  if (aodv_addressing_mode == InetC_Addr_Family_v4) {
    call aodv_rte_ip_datagram_create() to encapsulate rupdate_pkptr into
        an IP datagram for IPv4;
  }
  else {
    call aodv_rte_ip_datagram_create() to encapsulate rupdate_pkptr into
        an IP datagram for IPv6;

    /* Install the ICI for IPv6 case */
    ip_iciptr = op_ici_create ("ip_rte_req_v4");
    op_ici_attr_set (ip_iciptr, "multicast_major_port", mcast_major_port);
    op_ici_install (ip_iciptr);
  }

  set route_entry->last_route_expiry_update_time = curr_time;
  call manet_rte_to_cpu_pkt_send_schedule() to send the rupdate_pkptr packet;

  /* Clear the ICI if installed */
  op_ici_install (OPC_NIL);

  rupdate_sent_count++;
  add statistics;
}
```

C.3.1 aodv_pkt_support_option_mem_copy Function

To create a RUPDATE packet, the aodv_pkt_support_option_mem_copy() function is called by the aodv_pkt_support_pkt_create() function, which is called by aodv_rte_route_expiry_time_update_send() function. This function calls other functions to create different types of AODV control packets. Since RUPDATE is a new packet type, a new RUPDATE option is added to the aodv_pkt_support_option_mem_copy() function to call aodv_pkt_support_rupdate_option_create() (which is illustrated in Code C.3.2) to create a RUPDATE packet.

Code C.3.1: aodv_pkt_support_option_mem_copy() Function

```
AodvT_Packet_Option* aodv_pkt_support_option_mem_copy (option_ptr) {
  switch (option_ptr->type) {
    ...
    case (AODVC_ROUTE_UPDATE):
      set rupdate_option_ptr = (AodvT_Rupdate*) option_ptr->value_ptr;
      call aodv_pkt_support_rupdate_option_create() to
        create copy_option_ptr for the rupdate option;
      break;
    ...
  }
  return (copy_option_ptr);
}
```

C.3.2 aodv_pkt_support_rupdate_option_create Function

A RUPDATE packet is created using this new function.

Code C.3.2: aodv_pkt_support_rupdate_option_create() Function

```
AodvT_Packet_Option* aodv_pkt_support_rupdate_option_create (
                        pathchange_time, dest_addr) {
  /* Allocate memory for the route update option  **/
  rupdate_option_ptr = aodv_pkt_support_rupdate_option_mem_alloc();

  /* Set the variables of the rupdate option  */
  rupdate_option_ptr->pathchange_time = pathchange_time;
  rupdate_option_ptr->dest_addr = inet_address_copy (dest_addr);

  /* Allocate memory to set into the AODV packet option */
  aodv_pkt_option_ptr = aodv_pkt_support_option_mem_alloc();
  aodv_pkt_option_ptr->type = AODVC_ROUTE_UPDATE;
  aodv_pkt_option_ptr->value_ptr = (void*) rupdate_option_ptr;

  FRET (aodv_pkt_option_ptr);
}
```

C.3.3 aodv_pkt_support_rupdate_option_mem_alloc Function

This function is added to allocate memory for the RUPDATE packet.

Code C.3.3: aodv_pkt_support_rupdate_option_mem_alloc() Function

```
AodvT_Rupdate* aodv_pkt_support_rupdate_option_mem_alloc () {
  if (rupdate_option_pmh_defined == FALSE) {
    /* Call the pool memory handle for the route update option  */
    rupdate_option_pmh = op_prg_pmo_define ("AODV Route Update Option",
                                     sizeof (AodvT_Rupdate), 32);
    rupdate_option_pmh_defined = TRUE;
  }

  /* Allocate the route update option from the pooled memory  */
  rupdate_ptr = (AodvT_Rupdate*) op_prg_pmo_alloc (rupdate_option_pmh);

  return (rupdate_ptr);
  }
```

C.4 Process RUPDATE

Upon receiving a RUPDATE packet from a neighbouring node, the aodv_rte_pkt_arrival_handle() function is called, which is then call the aodv_rte_rupdate_pkt_arrival_handle() function (Code C.4) to process the RUPDATE packet. This function calls aodv_rte_expiry_time_update_process() (Code C.2.1) to check whether the node needs to update the route entry information, and whether it should broadcast a RUPDATE or not.

Code C.4: aodv_rte_rupdate_pkt_arrival_handle() Function

```
void aodv_rte_rupdate_pkt_arrival_handle (ip_pkptr, aodv_pkptr,
                   ip_dgram_field_ptr, ici_field_ptr, aodv_options_ptr) {

  /* Get the rupdate packet option from the arrived packet */
  set rupdate_pkptr = (AodvT_Rupdate*) tlv_options_ptr->value_ptr;

  /* Get the previous hop address of the rupdate   */
  /* packet, which is the source of the IP packet. */
  set prev_hop_addr = ip_dgram_field_ptr->src_addr;

  /* Predict linkchange and pathchange times */
  set linkchange_time = predict_linkchange_time (previous_hop_addr);
  set pathchange_time = min(rupdate_packet_ptr->pathchange_time, linkchange_time);

  if (linkchange_time < curr_time)
    set is_link_broken = TURE;

  /* Check if a RUPDATE is required to be sent from this node to the */
  /* precursors of the destination specified in the rupdate_pkptr.   */
  call aodv_route_table_entry_get() to get the route_entry for
      rupdate_pkptr->dest_addr in the RUPDATE packet from the route_table;

  if (route_entry exists &&
      route_entry->next_hop is prev_hop_addr) {
    call aodv_rte_expiry_time_update_process() to set an interrupt to send a
        RUPDATE or a RERR to the route_entry's precursor nodes if required;
  }

  /* Check if a RUPDATE is required to be sent from this node to */
  /* the precursors of the sender of the rupdate_pkptr (i.e. the */
  /* previous hop to the destination.                            */
  call aodv_route_table_entry_get() to get the route_entry for
      the prev_hop_addr from the route_table;

  if (route_entry exists &&
      rupdate_pkptr->dest_addr is not prev_hop_addr &&
      route_entry->next_hop is prev_hop_addr) {
    call aodv_rte_expiry_time_update_process() to set an interrupt to send
        RUPDATE or RERR to its precursor nodes if required;
  }
}
```

Appendix D

Parameters and Structures

This appendix contains information of the variables and parameters mentioned in this thesis.

D.1 Standard AODV Parameters

Table D.1 on page 190 consists of all the AODV parameters listed in the AODV standard [16].

Parameter	Value
ACTIVE_ROUTE_TIMEOUT	3,000 ms
ALLOWED_HELLO_LOSS	2
BLACKLIST_TIMEOUT	RREQ_RETRIES $\times$ NET_TRAVERSAL_TIME
DELETE_PERIOD	at least ALLOWED_HELLO_LOSS $\times$ HELLO_INTERVAL
HELLO_INTERVAL	1,000 ms
LOCAL_ADD_TTL	2
MAX_REPAIR_TTL	0.3 $\times$ NET_DIAMETER
MIN_REPAIR_TTL	last known hop count to the destination
MY_ROUTE_TIMEOUT	2 $\times$ ACTIVE_ROUTE_TIMEOUT
NET_DIAMETER	35
NET_TRAVERSAL_TIME	2 $\times$ NODE_TRAVERSAL_TIME $\times$ NET_DIAMETER
NEXT_HOP_WAIT	NODE_TRAVERSAL_TIME + 10
NODE_TRAVERSAL_TIME	40 ms
PATH_DISCOVERY_TIME	2 $\times$ NET_TRAVERSAL_TIME
RERR_RATELIMIT	10
RING_TRAVERSAL_TIME	2 $\times$ NODE_TRAVERSAL_TIME $\times$ (TTL_VALUE + TIMEOUT_BUFFER)
RREQ_RETRIES	2
RREQ_RATELIMIT	10
TIMEOUT_BUFFER	2
TTL_START	1
TTL_INCREMENT	2
TTL_THRESHOLD	7
TTL_VALUE	TTL field in the IP header

Table D.1: *AODV Parameters.*

D.2 Route Entry

A route entry structure, AodvT_Route_Entry, is used to store route information for a specific destination. This structure has been modified for AODV-PP and AODV-PU. The variables of the original AODV route entry structure are listed in Table D.2, the variables that are added to the AODV route entry structure for AODV-PP are listed in Table D.3 (on page 192), and the variables that are added to AODV-PP for AODV-PU are listed in Table D.4 (on page 193).

Variables	Descriptions
dest_prefix	IpT_Dest_Prefix of the destination, which can be converted to destination address.
dest_seq_num	Sequence number of the destination.
valid_dest_sequence_number_ flag	A flag that indicates whether the destination sequence number is valid.
route_entry_state	State of the route entry includes: UNDEFINED, VALID or INVALID.
next_hop_addr	Next hop address.
next_hop_port_info	Port information of the next hop address.
hop_count	Number of hops to the destination.
precursor_lptr	A list to store all the precursor nodes.
route_expiry_time	Route expiry time.
route_expiry_evhandle	Event handler for route expiry time interrupts.

Table D.2: *OPNET - Variables in struct AodvT_Route_Entry.*

Variables	Descriptions
rrep_send_timer	A timer set to send a RREP packet back to the source (which is the destination of the route entry). It is set when the destination node receives a RREQ packet.
rrep_send_timer_evhandle	An event handler for rrep_send_timer.
pre_route_start_timer	The time at which pre-route RREQ will be sent.
pre_route_start_timer_evhandle	Event handler for pre_route_start_timer.
pre_route_expiry_timer	The time when the pre-route process expires (i.e. stops waiting for more RREQs) and replace the route information with the pre-route information, and if the node is a destination, a RREP will be sent.
pre_route_expiry_timer_ evhandle	Event handler for pre_route_expiry_timer.
pre_route_dest_seq_num	Pre-route destination sequence number.
pre_route_valid_dest_sequence_ number_flag	Pre-route flag indicating if the destination sequence number is valid.
pre_route_next_hop_addr	Pre-route next hop address.
pre_route_next_hop_port_info	Pre-route port information for the next hop address.
pre_route_hop_count	Pre-route hop count to the destination.
pre_route_precursor_lptr	Pre-route precursor nodes list.
pre_route_route_expiry_time	Pre-route route expiry time.
pre_route_state	Pre-route state - TRUE if waiting for more RREQ; and FALSE otherwise.
is_source_node	TRUE if the current node sends application packets to the destination; and FALSE otherwise.
last_app_pkt_to_dest_time	The time of the last application packet sent to the destination.
last_pkt_to_dest_time	The time of the last data packet (either from this or other nodes) sent/forwarded to the destination.

Table D.3: *OPNET - Variables added to AODV-PP's AodvT_Route_Entry structure.*

Variables	Descriptions
last_route_expiry_update_time	The time at which the last RUPDATE packet was sent.
route_expiry_update_timer	The time to send a RUPDATE packet. = min(route_expiry_time, new_route_expiry_time) - 2 $\times$ MAX_RING_TRAVERSAL_TIME
route_expiry_update_evhandle	Event handler to handle interrupts for sending a RUPDATE packet.
new_route_expiry_time	The newly predicted route expiry time. This value will be used to update the route_expiry_time in Table D.2 upon sending a RUPDATE packet.
link_expiry_time	Link expiry time. (Used for debugging)
new_link_expiry_time	The newly predicted link expiry time. (Used for debugging)
pre_route_link_expiry_time	Pre-route link expiry time. (Used for debugging)

Table D.4: *OPNET - Variables added to AODV-PP's AodvT_Route_Entry structure.*

Bibliography

[1] S. Corson and J. Macker, "Mobile Ad hoc Networking (MANET): Routing protocol performance issues and evaluation considerations," RFC 2501, Jan, 1999. [Online]. Available: http://cabernet.tools.ietf.org/html/rfc2501

[2] S. J. Lee, W. Su, and M. Gerla, "Ad hoc wireless multicast with mobility prediction," in *Proceedings of the 8th International Conference on Computer Communications and Networks*. IEEE, 1999, pp. 4–9.

[3] W. Su, S. J. Lee, and M. Gerla, "Mobility prediction in wireless networks," in *21st Century Military Communications Conference Proceedings*, vol. 1. IEEE, 2000, pp. 491–495.

[4] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *International Journal of Network Management*, vol. 11, no. 1, pp. 3–30, 2001.

[5] S. Sharma, V. Alatzeth, G. Grewal, S. Pradhan, and A. Helmy, "A comparative study of mobility prediction schemes for GLS location service," in *IEEE 60th Vehicular Technology Conference (VTC)*, vol. 5. IEEE, 2004, pp. 3125–3129.

[6] N.-C. Wang and S.-W. Chang, "A reliable on-demand routing protocol for mobile ad hoc networks with mobility prediction," *Computer Communications*, vol. 29, no. 1, pp. 123–135, 2005.

[7] A. Agarwal and S. R. Das, “Dead reckoning in mobile ad hoc networks,” in *IEEE Wireless Communications and Networking Conference (WCNC)*, vol. 3. IEEE, 2003, pp. 1838–1843.

[8] M. Mamun-Or-Rashid and C. S. Hong, “LSLP: Link Stability and Lifetime Prediction based QoS aware routing for MANET,” in *Proceedings of the Joint Conference on Communications and Information (JCCI)*. Phoenix Park, Korea: JCCI, 2007, pp. 23–30.

[9] R. Bo, G. Amoussou, Z. Dziong, M. Kadoch, and A. K. Elhakeem, “Mobility prediction aided dynamic multicast routing in MANET,” in *IEEE Sarnoff Symposium on Advances in Wired and Wireless Communication*. IEEE, 2005, pp. 21–24.

[10] V. Kumar and S. R. Das, “Performance of dead reckoning-based location service for mobile ad hoc networks,” *Wireless Communications and Mobile Computing (WCMC)*, vol. 4, no. 2, pp. 189–202, 2004.

[11] J. Wang, Y. Tang, S. Deng, and J. Chen, “QoS routing with mobility prediction in MANET,” in *IEEE Pacific Rim Conference on Communications, Computers and signal Processing*, vol. 2. IEEE, 2001, pp. 357–360.

[12] W. Creixell and K. Sezaki, “Routing protocol for mobile ad hoc networks using mobility prediction,” in *2nd International Conference on Mobile Technology, Applications and Systems*. IEEE, 2005, pp. 1–6.

[13] F. D. Rosa, A. Malizia, and M. Mecella, “Disconnection prediction in mobile ad hoc networks for supporting cooperative work,” in *IEEE Pervasive Computing*, vol. 4. IEEE, 2005, pp. 62–70.

[14] T. Hossmann, “Mobility prediction in MANETs,” Ph.D. dissertation, ETH Zurich, 2006.

[15] K. Farkas, T. Hossmann, L. Ruf, and B. Plattner, “Pattern matching based link quality prediction in wireless mobile ad hoc networks,” in *Proceedings of the 9th ACM international symposium on Modeling*

analysis and simulation of wireless and mobile systems (MSWiM). Terromolinos, Spain: ACM, 2006, pp. 239–246.

[16] C. E. Perkins, E. M. Belding-Royer, and S. R. Das, "Ad hoc On-demand Distance Vector (AODV) routing," RFC 3561, July, 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txt

[17] S. Das, R. Castaneda, J. Yan, and R. Sengupta, "Comparative performance evaluation of routing protocols for mobile, ad hoc networks," in *Proceedings of the 7th International Conference on Computer Communications and Networks.* IEEE Computer Society, 1998, pp. 153–161.

[18] J. Broch, D. A. Maltz, D. B. Johnson, Y.-C. Hu, and J. Jetcheva, "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 (MobiCom).* Dallas, Texas, United States: ACM, 1998, pp. 85–97.

[19] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina, "Performance comparison of two on-demand routing protocols for ad hoc networks," *IEEE Personal Communications*, vol. 8, no. 1, pp. 16–28, 2001.

[20] X. Hong, M. Gerla, G. Pei, and C.-c. Chiang, "A group mobility model for ad hoc wireless networks," in *Proceedings of the 2nd ACM international workshop on Modeling, analysis and simulation of wireless and mobile systems (MSWiM).* Seattle, Washington, United States: ACM, 1999, pp. 53–60.

[21] E. Celebi, "Performance evaluation of wireless multi-hop ad-hoc network routing protocols," Masters, Bogazici University, 2002.

[22] S. Ahmed and M. S. Alam, "Performance evaluation of important ad hoc network protocols," *EURASIP Journal on Wireless Communications and Networking*, vol. 2006, no. 2, pp. 42–42, 2006.

[23] M. T. Hyland, B. E. Mullins, R. O. Baldwin, and M. A. Temple, "Simulation-based performance evaluation of mobile ad hoc routing protocols in a swarm of unmanned aerial vehicles," in *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW'07)*, vol. 02. IEEE Computer Society, 2007, pp. 249–256.

[24] A. B. R. Kumar, L. C. Reddy, and P. S. Hiremath, "Performance comparison of wireless mobile ad-hoc network routing protocols," *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 8, pp. 337–343, 2008.

[25] A. Neumann, C. Aichele, M. Lindner, and S. Wunderlich, "Better Approach To Mobile Ad-hoc Networking (B.A.T.M.A.N.) draft-wunderlich-openmesh-manet-routing-00," Internet-Draft, Apr, 2008. [Online]. Available: http://tools.ietf.org/html/draft-wunderlich-openmesh-manet-routing-00

[26] A. Koul, R. B. Patel, and V. K. Bhat, "Distance and frequency based route stability estimation in mobile adhoc networks," *Journal of Emerging Technologies in Web Intelligence*, vol. 2, no. 2, pp. 89–95, 2010.

[27] J. Moy, "OSPF version 2," RFC 2328, April 1998. [Online]. Available: http://tools.ietf.org/html/rfc2328

[28] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," in *SIGCOMM Proceedings of the conference on Communications architectures, protocols and applications*, vol. 24. London, England, UK: ACM, 1994, pp. 234–244.

[29] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC 3626, Oct, 2003. [Online]. Available: http://tools.ietf.org/html/rfc3626

[30] T. Clausen, C. Dearlove, P. Jacquet, and U. Herberg, "The Optimized

Link State Routing Protocol version 2," Internet-Draft, March, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-olsrv2-14

[31] D. Yadav and G. S. Mishra, "Performance evaluation of proactive routing protocol for ad-hoc networks," *International Journal of Computer Science and Information Technology and Security (IJCSITS)*, vol. 2, no. 3, pp. 690–695, 2012.

[32] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing protocol (DSR) for mobile ad hoc networks for IPv4," RFC 4728, Feb, 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4728.txt

[33] C. E. Perkins, S. Ratliff, and J. Dowdell, "Dynamic MANET On-demand (AODVv2) Routing," Internet-Draft, March, 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-dymo-22

[34] K. Tripathi, M. Pandey, and S. Verma, "Comparison of reactive and proactive routing protocols for different mobility conditions in wsn," in *Proceedings of the International Conference on Communication, Computing and Security*. ACM, 2011, pp. 156–161.

[35] S. Barakovic, B. Sarajevo, Herzegovina, and J. Barakovic, "Comparative performance evaluation of mobile ad hoc routing protocols," in *Proceedings of the 33rd International Convention (MIPRO)*. IEEE, 2010, pp. 518–523.

[36] C. Mbarushimana and A. Shahrabi, "Comparative study of reactive and proactive routing protocols performance in mobile ad hoc networks," in *21st International Conference on Advanced Information Networking and Applications Workshops*. IEEE, 2007.

[37] S. Hamma, E. Cizeron, H. Issaka, and J.-P. Guedon, "Performance evaluation of reactive and proactive routing protocol in IEEE 802.11 ad hoc network," in *Next-Generation Communication and Sensor Networks*. SPIE-The International Society for Optical Engineering, 2006.

[38] A. S. Tanenbaum, *Computer Networks (Third Edition).* Upper Saddle River, New Jersey: Prentice Hall, Inc., 1996.

[39] Z. J. Haas, M. R. Pearlman, and P. Samar, "The zone routing protocol (ZRP) for ad hoc networks," Internet-Draft, July 2002. [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-zone-zrp-04

[40] M. Gerla, X. Hong, and G. Pei, "Fisheye state routing protocol (FSR) for ad hoc networks," Internet-Draft, June 17, 2002. [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-fsr-03

[41] J. Ariyakhajorn, P. Wannawilai, and C. Sathitwiriyawong, "A comparative study of random waypoint and gauss-markov mobility models in the performance evaluation of MANET," in *International Symposium on Communications and Information Technologies (ISCIT).* IEEE, 2006, pp. 894–899.

[42] T. Camp, J. Boleng, and V. Davies, "A survey of mobility models for ad hoc network research," *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, vol. 2, no. 5, pp. 483–502, 2002.

[43] A. A. Agashe and S. K. Bodhe, "Performance evaluation of mobility models for wireless ad hoc networks," in *1st International Conference on Emerging Trends in Engineering and Technology (ICETET).* IEEE, 2008, pp. 172–175.

[44] D. Shukla and S. Iyer, "Mobility models in adhoc networks," 2001.

[45] D. Johnson and D. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing.* Kluwer Academic Publishers, 1996, pp. 153–181.

[46] A. Venkateswaran, V. Sarangan, N. Gautam, and R. Acharya, "Impact of mobility prediction on the temporal stability of MANET clustering algorithms," in *Proceedings of the 2nd ACM international workshop on Performance evaluation of wireless ad hoc, sensor, and ubiquitous*

networks (PE-WASUN). Montreal, Quebec, Canada: ACM, 2005, pp. 144–151.

[47] S. Gowrishankar, T. G. Basavaraju, and S. K. Sarkar, “Effect of random mobility models pattern in mobile ad hoc networks,” in *International Journal of Computer Science and Network Security (IJCSNS)*, vol. 7, No. 6, 2007, pp. 160–164.

[48] B. Liang and Z. J. Haas, “Predictive distance-based mobility management for pcs networks,” in *Proceedings of the 18th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3. IEEE, 1999, pp. 1377–1384.

[49] H. Xu, M. Meng, J. Cho, B. J. d’Auriol, and S. Lee, *Mobility Tracking for Mobile Ad Hoc Networks*, ser. Lecture Notes in Computer Science. Springer Berlin / Heidelberg, 2007*, vol. 4611/2007, pp. 285–294.

[50] J. Harri, C. Bonnet, and F. Filali, “The challenges of predicting mobility,” Institut Eurecom, Department of Mobile Communications, Tech. Rep. RR-06-171, 2007.

[51] S. H. Shah and K. Nahrstedt, “Predictive location-based QoS routing in mobile ad hoc networks,” in *IEEE International Conference on Communications (ICC)*, vol. 2. IEEE, 2002, pp. 1022–1027.

[52] K. Sithitavorn and Q. Bin, “Mobility prediction with direction tracking on dynamic source routing,” in *IEEE Region 10 TENCON.* IEEE, 2005, pp. 1–6.

[53] Q. J. Chen, S. S. Kanhere, M. Hassan, and K.-C. Lan, “Adaptive position update in geographic routing,” in *IEEE International Conference on Communications (ICC)*, vol. 9. IEEE, 2006, pp. 4046–4051.

[54] P. N. Pathirana, A. V. Savkin, and S. Jha, “Location estimation and trajectory prediction for cellular networks with mobile base stations,” *IEEE Transactions on Vehicular Technology*, vol. 53, no. 6, pp. 1903–1913, 2004.

[55] ——, “Mobility modelling and trajectory prediction for cellular networks with mobile base stations,” in *Proceedings of the 4th ACM international symposium on Mobile ad hoc networking and computing (MobiHoc)*. Annapolis, Maryland, USA: ACM, 2003, pp. 213–221.

[56] Z. R. Zaidi and B. L. Mark, “Real-time mobility tracking algorithms for cellular networks based on kalman filtering,” *IEEE Transactions on Mobile Computing*, vol. 4, no. 2, pp. 195–208, 2005.

[57] B. L. Mark and Z. R. Zaidi, “Robust mobility tracking for cellular networks,” in *IEEE International Conference on Communications (ICC)*, vol. 1. IEEE, 2002, pp. 445–449.

[58] J. Harri, C. Bonnet, and F. Filali, “Kinetic mobility management applied to vehicular ad hoc network protocols,” *Computer Communications*, vol. 31, no. 12, pp. 2907–2924, 2008.

[59] A. Venkateswaran, V. Sarangan, T. F. La Porta, and R. Acharya, “A mobility-prediction-based relay deployment framework for conserving power in MANETs,” *IEEE Transactions on Mobile Computing*, vol. 8, no. 6, pp. 750–765, 2009.

[60] J. Su, A. Chin, A. Popivanova, A. Goel, and E. d. Lara, “User mobility for opportunistic ad-hoc networking,” in *6th Workshop on Mobile Computing Systems and Applications*. UK: IEEE, 2004, pp. 41–50.

[61] Z. Shah and R. A. Malaney, “Particle filters and position tracking in Wi-Fi networks,” in *IEEE 63rd Vehicular Technology Conference (VTC)*, vol. 2. IEEE, 2006, pp. 613–617.

[62] Y. Gong, Y. Xiong, Q. Zhang, Z. Zhang, W. Wang, and Z. Xu, “WSN12-3: Anycast routing in delay tolerant networks,” in *IEEE Global Telecommunications Conference (GLOBECOM)*. IEEE, 2006, pp. 1–5.

[63] K. Itoh and H. Higaki, “Location estimation of wireless nodes with preservation of neighbor relations in MANET,” in *Proceedings of the 8th*

International Conference on Information Technology: New Generations (ITNG), S. Latifi, Ed. IEEE Computer Society, 2011, pp. 117–124.

[64] I. Guvenc, C. Abdallah, R. Jordan, and R. Jordan, “Enhancements to RSS based indoor tracking systems using kalman filters,” in *International Signal Processing Conference (ISPC) and Global Signal Processing Expo (GSPx)*, Dallas, TX, USA, 2003.

[65] K. Sayrafian-Pour and J. Perez, “Robust indoor positioning based on received signal strength,” in *2nd International Conference on Pervasive Computing and Applications*. IEEE, 2007, pp. 693–698.

[66] E. Elnahrawy, X. Li, and R. Martin, “The limits of localization using signal strength: a comparative study,” in *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks*. IEEE, 2004, pp. 406–414.

[67] S. Chitte, S. Dasgupta, and Z. Ding, “Distance estimation from received signal strength under log-normal shadowing: Bias and variance,” *IEEE Signal Processing Letters*, vol. 16, no. 3, pp. 216–218, 2009.

[68] D. Wu, Y. Zhen, C. Xu, M. Wu, and J. Liu, “On-demand reliable routing mechanism for MANET based on link lifetime predicting,” in *4th International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*. IEEE, 2008, pp. 1–4.

[69] K. Paul, S. Bandyopadhyay, A. Mukherjee, and D. Saha, “Communication aware mobile hosts in ad-hoc wireless network,” in *International Conference on Personal Wireless Communications*. IEEE, 1999, pp. 83–87.

[70] S. Agarwal, A. Ahuja, J. P. Singh, and R. Shorey, “Route-lifetime assessment based routing (rabr) protocol for mobile ad-hoc networks,” in *IEEE International Conference on Communications (ICC)*, vol. 3. IEEE, 2000, pp. 1697–1701.

[71] S. Jiang, "An enhanced prediction-based link availability estimation for MANETs," *IEEE Transactions on Communications*, vol. 52, no. 2, pp. 183–186, 2004.

[72] S. Jiang, D. He, and J. Rao, "A prediction-based link availability estimation for mobile ad hoc networks," in *Proceedings of the 20th Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM)*, vol. 3. IEEE, 2001, pp. 1745–1752.

[73] ——, "A prediction-based link availability estimation for routing metrics in MANETs," *IEEE/ACM Transactions on Networking*, vol. 13, no. 6, pp. 1302–1312, 2005.

[74] H. Kaaniche and F. Kamoun, "Mobility prediction in wireless ad hoc networks using neural networks," *Journal of Telecommunications*, vol. 2, no. 1, pp. 95–101, 2010.

[75] L. Gong, Y. Bai, M. Chen, and D. Qian, "Link availability prediction in ad hoc networks," in *14th IEEE International Conference on Parallel and Distributed Systems*. IEEE, 2008, pp. 423–428.

[76] E. Y. Hua and Z. J. Haas, "An algorithm for prediction of link lifetime in MANET based on unscented kalman filter," *IEEE Communications Letters*, vol. 13, no. 10, pp. 782–784 784, 2009.

[77] Z. Li, L. Sun, and E. C. Ifeachor, "Range-based mobility estimations in MANETs with application to link availability prediction," in *IEEE International Conference on Communications (ICC)*. IEEE, 2008, pp. 3376–3382.

[78] Q. Han, Y. Bai, L. Gong, and W. Wu, "Link availability prediction-based reliable routing for mobile ad hoc networks," *IET Communications*, vol. 5, no. 16, pp. 2291–2300, 2011.

[79] S. Kwon, H. Park, and K. Lee, "A novel mobility prediction algorithm based on user movement history in wireless networks," in *Proceedings of the 3rd Asian simulation conference on Systems Modeling and Simulation:*

theory and applications (AsiaSim). Springer-Verlag Berlin, Heidelberg, 2004, pp. 419–428.

[80] F. Erbas, J. Steuer, D. Eggesieker, K. Kyamakya, and K. Jobinann, “A regular path recognition method and prediction of user movements in wireless networks,” in *IEEE 54th Vehicular Technology Conference (VTC)*, vol. 4. IEEE, 2001, pp. 2672–2676.

[81] I.-R. Chen and V. N., “Simulation study of a class of autonomous host-centric mobility prediction algorithms for wireless cellular and ad hoc networks,” in *36th Annual Simulation Symposium.* IEEE, 2003, pp. 65–72.

[82] J.-M. Gil, C. Y. Park, C.-S. Hwang, D.-S. Park, J. G. Shon, and Y.-S. Jeong, “Restoration scheme of mobility databases by mobility learning and prediction in pcs networks,” *IEEE Journal on Selected Areas in Communications*, vol. 19, no. 10, pp. 1962–1973, 2001.

[83] C. Mala, M. Loganathan, N. P. Gopalan, and B. SivaSelvan, *A Novel Genetic Algorithm Approach to Mobility Prediction in Wireless Networks.* Spinger, 2009, vol. 40, pp. 49–57.

[84] Z. R. Zaidi and B. L. Mark, “Mobility estimation based on an autoregressive model,” in *IEEE Transactions on Vehicular Technology.* Electrical Engineering Department of George Mason University, USA, Tech. Rep., 2004.

[85] ——, “Distributed mobility tracking for ad hoc networks based on an autoregressive model,” in *Proceedings of the 6th international conference on Distributed Computing (IWDC)*, ser. Lecture Notes in Computer Science, Session VI: Ad Hoc Networks, vol. 3326. Kolkata, India: Springer-Verlag Berlin, Heidelberg, 2005, pp. 447–458.

[86] E. R. Cavalcanti and M. A. Spohn, “Predicting mobility metrics through regression analysis for random, group, and grid-based mobility models in MANETs,” in *IEEE Symposium on Computers and Communications (ISCC).* IEEE, 2010, pp. 443–448.

[87] R. R. Sarukkai, "Link prediction and path analysis using markov chains," *Computer Networks*, vol. 33, no. 1-6, pp. 377–386, 2000.

[88] S. K. Hwang and D. S. Kim, "Markov model of link connectivity in mobile ad hoc networks," *Telecommunication Systems*, vol. 34, no. 1-2, pp. 51–58, 2007.

[89] S. Mousavi, H. Rabiee, M. Moshref, and A. Dabirmoghaddam, "Model based adaptive mobility prediction in mobile ad-hoc networks," in *International Conference on Wireless Communications, Networking and Mobile Computing (WiCOM)*. Shanghai: IEEE, 2007, pp. 1713–1716.

[90] F. Gustafsson, F. Gunnarsson, N. Bergman, U. Forssell, J. Jansson, R. Karlsson, and P. J. Nordlund, "Particle filters for positioning, navigation, and tracking," *IEEE Transactions on Signal Processing*, vol. 50, no. 2, pp. 425–437, 2002.

[91] L. Mihaylova, D. Angelova, S. Honary, D. R. Bull, C. N. Canagarajah, and B. Ristic, "Mobility tracking in cellular networks using particle filtering," *IEEE Transactions on Wireless Communications*, vol. 6, no. 10, pp. 3589–3599, 2007.

[92] S.-C. Liou and Y.-M. Huang, "Trajectory predictions in mobile networks," *International Journal of Information Technology*, vol. 11, no. 11, pp. 109–122, 2005.

[93] J. Capka and R. Boutaba, "Mobility prediction in wireless networks using neural networks," in *Proceedings of the IFIP/IEEE international Conference on the Management of Multimedia Networks and Services (MMNS)*, vol. 3271. Springer Berlin / Heidelberg, 2004, pp. 320–333.

[94] L. Mihaylova, D. Bull, D. Angelova, and N. Canagarajah, "Mobility tracking in cellular networks with sequential monte carlo filters," in *8th International Conference on Information Fusion*, vol. 1. IEEE, 2005, p. 8.

[95] Z. R. Zaidi and B. L. Mark, "Mobility estimation for wireless networks based on an autoregressive model," in *IEEE Global Telecommunications Conference (GLOBECOM)*, vol. 6. IEEE, 2004, pp. 3405–3409.

[96] P. Shoubridge, M. Kraetzl, W. Wallis, and H. Bunke, "Detection of abnormal change in a time series of graphs," *Journal of Interconnection Networks*, vol. 3, no. 1-2, pp. 85–101, 2002.

[97] R. Chellappa-Doss, A. Jennings, and N. Shenoy, "A review of current mobility prediction techniques for ad hoc networks," in *Proceedings of the 4th IASTED International Multi-Conference on Wireless and Optical Communications*. Banff, Canada: ACTA Press, 2004.

[98] "MASON (Multi-Agent Simulator Of Neighborhoods)," Evolutionary Computation Laboratory of George Mason University, 2003. [Online]. Available: http://cs.gmu.edu/~eclab/projects/mason/

[99] "Matlab and simulink," The MathWorks, Inc., 1994. [Online]. Available: http://www.mathworks.com/

[100] R. Mathivaruni and V.Vaidehi, "An activity based mobility prediction strategy using markov modeling for wireless networks," in *Proceedings of the World Congress on Engineering and Computer Science (WCECS)*. San Francisco, USA: International Association of Engineers, 2008.

[101] M. K. Denko, *Mobility Prediction Schemes in Wireless Ad Hoc Networks*. Springerlink, 2005, vol. 26, pp. 171–186.

[102] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Transaction of the ASME: Journal of Basic Engineering*, vol. 82, no. D, pp. 35–45, 1960.

[103] G. Welch and G. Bishop, "An introduction to the kalman filter," University of North Carolina at Chapel Hill, Technical Report TR 95-041, 24 July 2006.

[104] C. E. Perkins, *Ad Hoc Networking*. Upper Saddle River, New Jersey: Addison-Wesley, 2000.

[105] "OPNET," OPNET Technologies, Inc., 1986. [Online]. Available: http://www.opnet.com/

[106] "Network simulator NS-2," 1989. [Online]. Available: http://www.isi.edu/nsnam/ns/

[107] "Qualnet," SCALABLE Network Technologies, Inc. [Online]. Available: http://www.scalable-networks.com

[108] "OMNeT++." [Online]. Available: http://www.omnetpp.org/

[109] B. Schilling, "Qualitative comparison of network simulation tools," Institute of Parallel and Distributed Systems (IPVS), University of Stuttgart, Tech. Rep., 31st January 2005.

[110] N. Sultana, N. Ullah, H. Kabir, N. Sultana, and K. S. Kwak, "A case study of networks simulation tools for wireless networks," in *Third Asia International Conference on Modelling and Simulation*. IEEE, 2009.

[111] S. Siraj, A. K. Gupta, and Rinku-Badgujar, "Network simulation tools survey," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 1, no. 4, pp. 201–210, 2012.

[112] J. Pan, "A survey of network simulation tools: Current status and future developments," Washington University, Tech. Rep., 2008.

[113] "OPNET software problem report," Report ID 153334, 30 Jul 2011. [Online]. Available: https://enterprise1.opnet.com/spr/4dcgi/SPRCLT_ViewSPR?ViewSPR_SPRID=153334

[114] "OPNET software problem report," Report ID 151780, 20 Jun 2011. [Online]. Available: https://enterprise1.opnet.com/spr/4dcgi/SPRCLT_ViewSPR?ViewSPR_SPRID=151780

[115] A. Al Hanbali, E. Altman, and P. Nain, "A survey of TCP over ad hoc networks," *IEEE Communications Surveys and Tutorials*, vol. 7, no. 3, pp. 22–36, 2005.

[116] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP vegas: New techniques for congestion detection and avoidance," in *Proceedings of the conference on Communications architectures, protocols and applications*. ACM, 1994, pp. 24–35.

[117] G. Xylomenos, G. C. Polyzos, P. Mahonen, and M. Saaranen, "TCP performance issues over wireless links," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52–58, 2001.