Design and Performance Analysis of an Aeronautical Routing Protocol with Ground Station Updates

Hemanth Narra

Submitted to the graduate degree program in Electrical Engineering & Computer Science and the Graduate Faculty of the University of Kansas School of Engineering in partial fulfillment of the requirements for the degree of Master of Science

Thesis Committee:

Dr. James P.G. Sterbenz: Chairperson

Dr. Gary Minden

Dr. Prasad Kulkarni

Date Defended

The Thesis Committee for Hemanth Narra certifies
that this is the approved version of the following thesis:

Design and Performance Analysis of an Aeronautical Routing Protocol with Ground Station Updates

Committee:

Dr. James P.G. Sterbenz: Chairperson

Dr. Gary Minden

Dr. Prasad Kulkarni

Date Approved

Abstract

Aeronautical routing protocol (AeroRP) is a position-based routing protocol developed for highly dynamic airborne networks. It works in conjunction with the aeronautical network protocol (AeroNP). AeroRP is a multi-modal protocol that operates in different modes depending on the mission requirements. Ground station (GS) update mode is an AeroRP mode in which the GS sends geolocation or topology updates to improve routing accuracy. The main contribution of this thesis is to develop and implement the GS updates in AeroRP and analyse its performance in the various modes and compare them against canonical MANET routing protocols such as DSDV, OLSR, AODV, and DSR. The simulation analysis shows that AeroRP outperforms the traditional MANET protocols in various scenarios.

I like to dedicate this work to my parents and my sister for their continuous support and guidance with which I could reach this level.

Acknowledgements

I would like to thank my committee members, especially my advisor Dr. James P.G. Sterbenz for his continuous support and encouragement. I would also like to thank PhD students Abdul Jabbar, Justin Rohrer and Egemen Çetinkaya for mentoring and advising me during the design and implementation of this protocol. The ResiliNets team has supported me a lot during my research and I would like to thank each and everyone of them for their support. Last but not the least I would like to thank my family most importantly my father Sambasiva Rao Narra, mother Jhansi Lakshmi Narra and sister Neeharika Narra, and all my friends and well-wishers for their support and guidance.

Contents

List of Figures

List of Tables

Chapter 1

Introduction and Motivation

Over the past decade there has been a tremendous growth in the usage of mobile wireless devices such as laptops, cell phones, and net-books. Connectivity among these devices is normally accomplished by taking services from fixed network infrastructure and centralised administration. However, in environments where there is no support of fixed network infrastructure or during emergency operations at places where the fixed infrastructure is broken, the devices cannot communicate with each other. Networks that can operate in such environments are known as mobile ad hoc networks (MANETs) [7] [8].

MANETs are self configuring wireless networks with mobile nodes. Unlike the conventional wired networks, MANETs do not have the support of fixed network infrastructure. In MANETs, nodes in addition to performing their usual tasks will also act as routers and communicate among themselves to form an ad hoc network. Over the course of their interaction, they exchange control messages for administrative functions along with data messages. Some nodes piggy-back the control messages along with the data messages to reduce control overhead and packet collisions. Network topologies of MANETs change dynamically due to

continuous movement of nodes. MANETs are mostly used in shielded or remote environments, at times of a natural disaster where (re)deployment of infrastructure is infeasible, and during military operations. These networks typically employ multi-hop routing protocols to discover end-to-end paths from a source to destination. In some cases the routing protocols may just determine the best next hop neighbour and leave the decision of choosing a path completely to this next hop node. The latter is generally preferred in case of a highly dynamic topology where nodes move at very high velocities.

The present day airborne telemetry architecture uses legacy point-to-point links connecting multiple sources (airborne nodes) to a ground station. The increased usage of wireless devices to meet the emerging needs of Major Range and Test Facility Bases (MRTFB) led to increased requirements for bandwidth and connectivity. These legacy point-to-point links will not be able to cope with the limited spectrum. This need is recognised by various groups including the Integrated Network Enhanced Telemetry (iNET) group [9], [10]. Multihop routing protocols are necessary for the airborne airborne nodes (ANs) to operate as an integrated system. ANs in these environments move at velocities of up to Mach 3.5 and move towards each other with relative velocities of about Mach 7.0. These high velocities lead to frequent link breaks and inconsistent routing of packets among nodes.

The aeronautical routing protocol (AeroRP) is one such domain specific routing protocol that is designed for highly dynamic airborne networks [11]. AeroRP is first introduced in [12] and later modelled and analysed using the ns-3 network simulator [13, 14]. The preliminary results showed that AeroRP outperforms the traditional MANET routing protocols in terms of throughput and packet delivery

ratio (PDR) [13,14]. AeroRP uses the node's current or predicted geolocation[1] information for discovering routes. Neighbour discovery in AeroRP is multi-modal, in which various modes can be used to discover neighbours depending upon the mission needs including stealth requirements. We will describe in detail the various operational modes of AeroRP in Section 2.2.1.

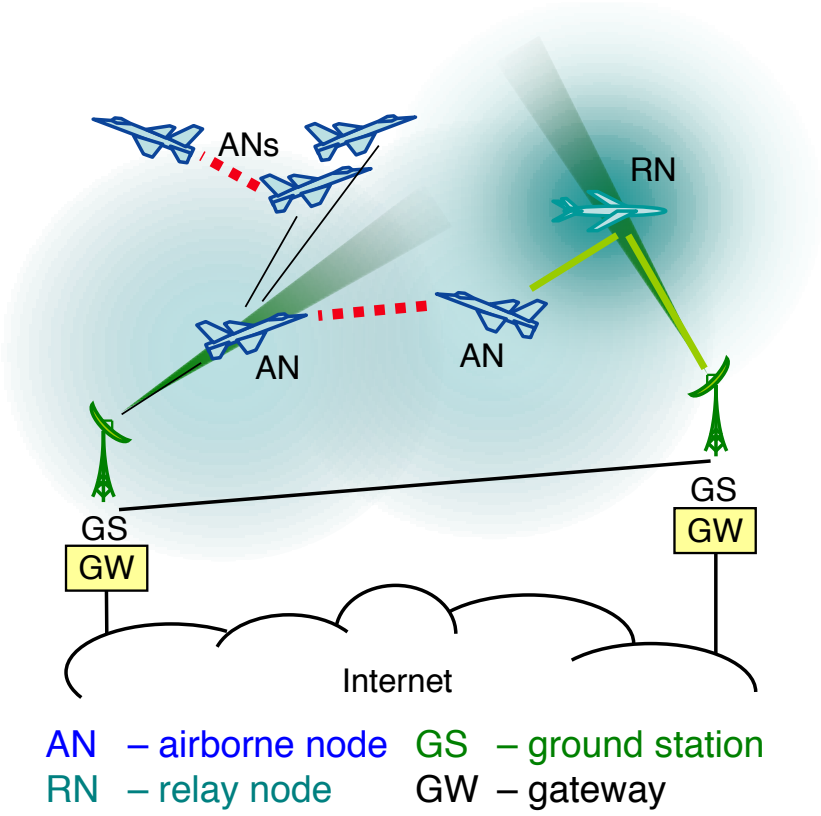

Figure 1.1. Dynamic airborne tactical environment [1]

1.1 Problem Statement

The typical environment of a highly dynamic airborne network is shown in Figure 1.1. Every airborne network has at least one *ground station* (GS), a set

[1]The words geolocation and position can be used interchangeably and mean the same in the context of this thesis.

airborne nodes (ANs) and optional *relay nodes* (RNs). Our assumption is that the GS knows geolocation and velocity information of every other AN or RN in the network. The usual traffic pattern for this environment is that the ANs send application data to the GS and the GS sends control information back to the ANs. The RNs are used to relay information to and from the GS to other ANs. The GS and an AN can also communicate directly with each other or via the RNs or other ANs, if it is beyond the transmission range of the GS. AeroRP in [6], can operate in two modes depending on the AN update mechanism: beacon and beaconless. In the beacon mode, ANs exchange geolocation information by transmitting periodic hello messages and in the beaconless mode, they piggy-back this information to data packets. Exchange of this information among nodes is only possible if they are within transmission range of each other. However, due to the highly dynamic nature of this environment, nodes stay connected only for a very short duration, possibly as low as 10 s. The ANs do not have geolocation information of every other node in the network except the nodes they communicated with previously. Lack of this information can affect the forwarding decisions made by a node. As we have already seen, there is least one GS in every network that tracks every node in the network and keeps a record of their geolocation information. However, this information is not shared with the other nodes in [6]. Sharing this information with other nodes is critical if the ANs are supposed to follow a pre-determined flight plan.

In addition, exchange of geolocation information is not a viable option in all environments; it reveals the exact position of a node. This can lead to problematic situations if this geolocation information is snooped by a malicious node. AeroRP should have a neighbour discovery mode that does not reveal the node's

geolocation information but inform it about its neighbours.

1.2 Proposed Solution

The GS has geolocation information of all the nodes in the network. We propose to leverage this information and pass it on to the other nodes. In this case all nodes will have geolocation information of every other node, thus allowing them to make better forwarding decisions. Controlled flooding is used to push these updates from the GS for every *periodic update interval*.

To overcome the security concerns or mission requirements of not exposing a node's geolocation information, we propose to broadcast the network topology information in the form of link state information between nodes. This topology information will just reveal the nodes on either side of a link and the link duration specified by a start time and an end time and the link cost. This information is sent from the GS for every *periodic update interval*. Also, based on the geolocation information of the node and its velocity component, the GS can predict the future topology and transmit that information as well. This further improves the node's ability to discover forwarding nodes.

The goal of this research is to modify the existing AeroRP protocol so it can make better forwarding decisions with the entire topology information at its disposal and design a new neighbour discovery mode that does not reveal the node's geolocation information. To accomplish the above solutions, the AeroRP message header formats [6] should be modified and new headers should be designed to transmit GS updates. AeroNP, an IP compatible network protocol should be implemented so that it could carry the geolocation information of the nodes and provide QoS, congestion-control, and error-detection services to the AeroTP

transport protocol. AeroRP will then be extensively simulated comparing its performance to the existing MANET protocols.

1.3 Contributions

The contributions of this thesis are the following:

- Design the AeroRP message headers and model the protocol for improving its performance
  - Choose the protocol parameters that can be used to modify the protocol operation
  - Incorporate `GS` updates and device a methodology of broadcasting these updates
  - Modify the protocol to use geolocation and topology information broadcasted by the GS in making routing decisions
- Implement AeroNP network protocol in ns-3
  - Modify AeroNP headers to suit the implementation decisions
  - Implement the QoS and congestion control services provided by the AeroNP protocol
- Implement the GS update mode and location-unaware routing in AeroRP routing protocol and the AeroNP network protocol in ns-3 network simulator
- Implement DSDV routing protocol in ns-3 to compare against AeroRP
- Implement TDMA MAC protocol in ns-3 over a simple-wireless channel model

- Analyse the performance of AeroRP in its various modes of operation and compare its performance against other MANET routing protocols such as OLSR, AODV, DSDV, and DSR in ns-3

1.4 Organisation

The rest of the thesis is organised as follows. Chapter 2 briefly discusses the classification of MANET routing protocols by providing examples from some of the canonical MANET routing protocols. It also describes AeroRP and AeroNP. Implementation details of DSDV and TDMA in ns-3 are outlined in Chapter 3. AeroRP with GS updates in explained in detail in Chapter 4 along with the its implementation in ns-3. Chapter 5 explains the AeroNP protocol and its implementation in ns-3. The simulations of the routing protocols and analysis of results in detailed in Chapter 6. Chapter 7 presents the conclusions and details the focus areas for future work.

Chapter 2

Background and Related Work

Routing protocols operate in the network layer of the protocol stack and discover paths between a source and a destination. The discovered paths are then populated in the node's forwarding tables. When a packet arrives at an intermediate node and destined for a particular destination, the intermediate node refers to its forwarding table to determine the next hop address for that destination. The packet is then forwarded to that next hop node. Routing protocols use routing algorithms to discover paths. There are many routing protocols developed for MANETs [15] [16] [17]. Development of MANET routing protocols continues to be an active research area to date as no single routing protocol is able to address all the challenges posed by ad-hoc networks.

This chapter is organised as follows. Section 2.1 describes the different types of routing protocols designed for MANET environment. Sections 2.1.1 and 2.1.2 discusses the various *topology-based* and *position-based* routing protocols, their features, and briefly elaborates on some of the canonical routing protocols from each category. This is followed by comparing the topology-based and the position-based routing protocols and explaining the advantages of the latter. Section 2.2.1 de-

scribes the AeroRP routing protocol without ground station advertisements (GSAs) and Section 2.2.2 briefly discusses the IP compatible AeroNP network protocol.

2.1 MANET Routing Protocols

The primary features of MANETs such as mobility and lack of infrastructure-support, pose a significant challenge to accurate routing of packets [18] [19]. Thus the protocols being designed for MANETs should take these effects into consideration. Routing protocols are classified as topology-based and position-based based on the type of information used for discovering routes. Topology-based protocols use information about the existing links among nodes whereas position-based protocols use the geographic position of nodes to perform packet forwarding. Topology-based routing protocols are further classified as *proactive*, *reactive*, and *source routing* protocols. Position-based protocols are classified into *greedy packet forwarding*, *restricted directional flooding*, and *hierarchical* routing protocols. These classifications are explained in detail in the following sections.

2.1.1 Topology-Based Routing Protocols

Topology-based routing protocols operate by identifying neighbours or existing link-state information, and exchanging this with other nodes in the network. Topology-based routing protocols are classified as proactive, reactive routing protocols based on the type of route discovery mechanism. Source routing is orthogonal to both reactive and proactive classification. It is a route discovery mechanism that can be classified either as a proactive or a reactive mechanism. Dynamic source routing (DSR) is a good example of a source routing protocol that can be classified as a reactive routing protocol as well. Subsequent sections

elaborate more on proactive, reactive, and source routing routing protocols by taking examples from the prominent topology-based routing protocols.

2.1.1.1 Proactive Routing Protocols

Proactive routing protocols maintain routes to all nodes in the network even if there is no request for a route. They add new routes or update existing routes by periodically distributing routing tables or exchanging link-state information with each other. One advantage of doing so is that routes to any destination are ready for use if needed. Some of the canonical proactive routing protocols are DSDV and OLSR.

DSDV: Destination-sequenced distance vector routing protocol [20] uses the Bellman-Ford algorithm to calculate paths. The cost metric used is the *hop count*, that is the number of hops it takes for the packet to reach its destination. DSDV is a table-driven proactive protocol, thus it maintains a routing table with entries for all the nodes in the network and not just the neighbours of a node. The changes are propagated through periodic and trigger update mechanisms used by DSDV. Because of these updates, the chances of having routing loops within a network increases. To eliminate routing loops, each update from the node is tagged with a *sequence number*. A sequence number for each node is independently chosen, but it must be incremented each time a periodic update is made by a node. The sequence number of normal update must be an even number, since each time a periodic update is made the node increments its sequence number by 2 and adds its update to the routing message it transmits. The node cannot change the sequence number of other nodes. Only if a node wants to send an update for an expired route to its neighbours, it increments the sequence number of the disconnected

node by one. Nodes receiving this update look at the sequence number and if it is odd, they remove the corresponding entry from the routing table. Mobility of the nodes in MANETs causes route fluctuations, for which DSDV uses *settling time* to dampen.

OLSR: Optimised link state routing [21] is a proactive routing protocol in which routes to all destinations within the network are discovered and maintained before a packet is sent from source to destination. OLSR uses HELLO and topology control (TC) messages to discover and broadcast *link state* information throughout the network regularly. Nodes receiving this topology information compute next hop destinations for all nodes in the network. HELLO messages at each node discover 2-hop neighbour information and select a set of multi-point relays (MPRs). MPRs are responsible for transmitting broadcast messages and constructing link state. OLSR floods topology data frequently enough over the network to make sure all nodes are synchronised with link state information.

2.1.1.2 Reactive Routing Protocols

Reactive routing protocols discover routes only if required. Nodes using reactive routing protocols will not update their routing tables periodically and will not maintain routes to all nodes in the network. Reactive routing protocols initiate a route request message to discover new routes if required. The main drawback of these protocols is the delay in discovering routes to new destinations. AODV is the most well-known reactive routing protocol.

AODV: Ad hoc on-demand distance vector [22, 23] is a distance vector routing protocol that operates reactively to reduce overhead finding routes only on de-

mand. If a route to a given destination does not exist, a route request (RREQ) message is flooded by the source and by the intermediate nodes if they have no previous routes in their table. Upon receiving a RREQ message, the receiving node records the route information in its own routing table. Once the RREQ message reaches the destination or an intermediate node, the node responds by unicasting a route reply (RREP) message back to the neighbour from where it first received the RREQ message. As the RREP message is forwarded back along the reverse path, nodes along this path set up forwarding entries in their routing tables, pointing to the node from where they received RREP message. AODV uses sequence numbers created by the destination for every route entry to avoid routing loops. Routes with the largest sequence number are preferred in selecting routes from the source to the destination.

2.1.1.3 Source Routing Protocols

In *source routing*, the sender node specifies partial or entire route traversed by the packet. This is different to many MANET routing protocols in which the next hop neighbour is free to choose any path to the destination.

DSR: Dynamic source routing [24] [25] is an on-demand routing protocol that employs the source-routing mechanism instead of route-request mechanism employed by AODV. Route discovery and maintenance are the two major phases in DSR operation. DSR maintains a route cache containing source routes to every other node in the network. If a source node wants to send a packet to a destination node, DSR on the source node looks for a route to destination in the route cache. If a route is identified, it adds the source routes to the packet and forwards the packet. The packet traverses all the nodes in the path specified by

the source route till it reaches the destination. If a route can not be determined using the route cache, the source node initiates a route discovery process and sends a RouteRequest message to every node within its transmission range. Nodes receiving the RouteRequest message look at their route-cache to see if they have a route to that destination. If they can not determine a route, they add their IP address to the RouteRequest message and broadcast it again. A route record is formed as this RouteRequest message is propagated through the network. When the RouteRequest reaches the destination, it replies back with a RouteReply message. The route record from the RouteRequest message is copied to a RouteReply message. The RouteReply message traverses the path specified by the route record to reach the source node. On receiving the RouteReply message, the source node as well as the intermediate nodes update their route cache with this route record. The sender node adds source route information to data packet and it traverses the path specified by it. Every node along the path is responsible for making sure the packet has reached the next hop along the source route. If any intermediate node does not receive the ReceiptRequest from the next hop, it should retransmit the packet. If the retransmission number reaches a maximum count, a RouteError message is sent to the sender node. The sender node then removes this broken link and uses an alternate entry from its route-cache. If necessary, it starts route discovery process again to find a route to destination.

2.1.2 Position-Based Routing Protocols

Position-based routing protocols use the geographic position information of a node in making forwarding decisions [26] [27]. The GPS receiver is commonly used to get geolocation and velocity information of a node. Unlike *topology-based*

protocols, *position-based* protocols do not require establishment or maintenance of routes. All forwarding decisions are made based on the current position of the destination and the source node's immediate neighbours. Based on the forwarding strategy employed by a routing protocol, packets are either forwarded to immediate neighbours closer to the destination, or closer to the source, or to all neighbours within a particular region. A discussion of the forwarding strategies and routing protocols employing them is presented below.

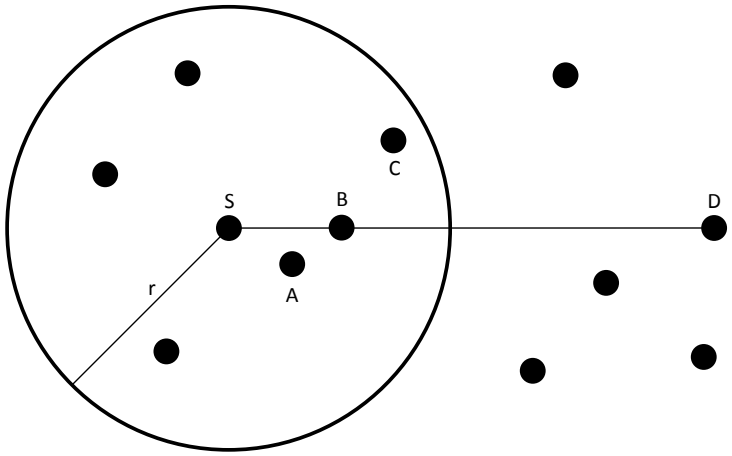

Figure 2.1. Greedy routing strategies (adapted from [2])

2.1.2.1 Greedy Packet Forwarding

In greedy packet forwarding, the sender node forwards packets to its neighbour node in the direction of destination. The criteria for choosing a forwarding neighbour can vary based on the forwarding strategy chosen by a node. In Figure 2.1, source S wants to send a packet to destination D. The transmission range of S is marked with a circle of radius r and D is outside the transmission range of S. Source S needs to forward the packet to one of its neighbours within the

transmission range so that they could forward the packet to D. One strategy, known as *most-forward within r* (MFR) [28] forwards the packet to a neighbour closest to destination, in this case C. MFR minimises the number of hops taken by packets to reach D. *Nearest with forward progress* (NFP) [29] is another strategy in which a source node forwards the packets to its closest neighbour by reducing its transmission power. In this case, source node S forwards the packets to neighbour A. Packet collisions can be significantly reduced using NFP strategy as packets stay for shorter durations on the wireless links. In other strategy known as *compass routing*, a packet is forwarded to a node that is on a closest angle to the destination. Compass routing eliminates the traversal of packet only in forward direction towards the destination. This feature allows it to successfully route a packet through a complex boundary even though there is no direct path to a destination.

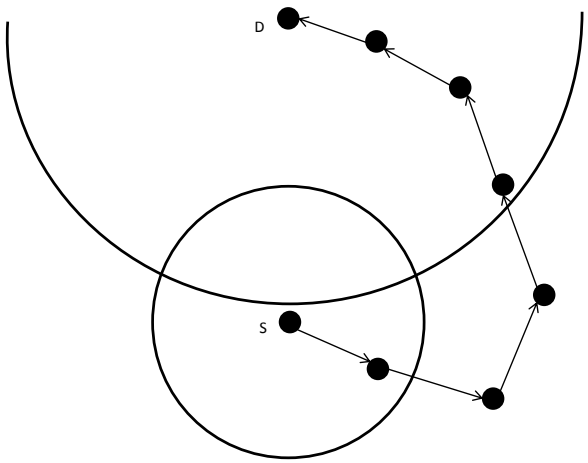

Figure 2.2. Greedy routing strategies (adapted from [2])

Greedy packet forwarding has many disadvantages such as routing loops and

failing to find a path between source and destination even if there exists one. Consider the scenario depicted in Figure 2.2, in which source S wants to find a route to destination D. Transmission range of S is depicted by the circle around it and the semicircle represents the distance between source S and destination D. A valid path exists between S and D. However, S using a greedy forwarding approach, it does not forward the packet using the existing path as it is closer to the destination than any other node in its transmission range.

2.1.2.2 Restricted Directional Flooding

Restricted directional flooding is a routing strategy in which a node forwards packets to more than one next hop neighbours in the forward direction. This strategy may increase the chances of packet reaching the destination, but cannot not guarantee its reception by the destination. In this section we look at two protocols employing this strategy, DREAM and LAR.

DREAM: Distance routing effect algorithm for mobility [3] is a geographic routing protocol employing restricted directional flooding strategy. Each node maintains a database for storing location information of every other node in the network. GPS is used to identify their current location. DREAM provides a novel approach for dissemination of its location information to other nodes. This approach is based on a simple observation called *distance-effect* that says, the greater the distance between two nodes, the slower they appear to be moving. Thus, the update frequency of location information to a node farther away can be reduced compared to a node that is much closer. The mobility rate is another factor that affects the update frequency, the faster a node moves, the more often it needs to be communicated with. DREAM uses the above two factors and

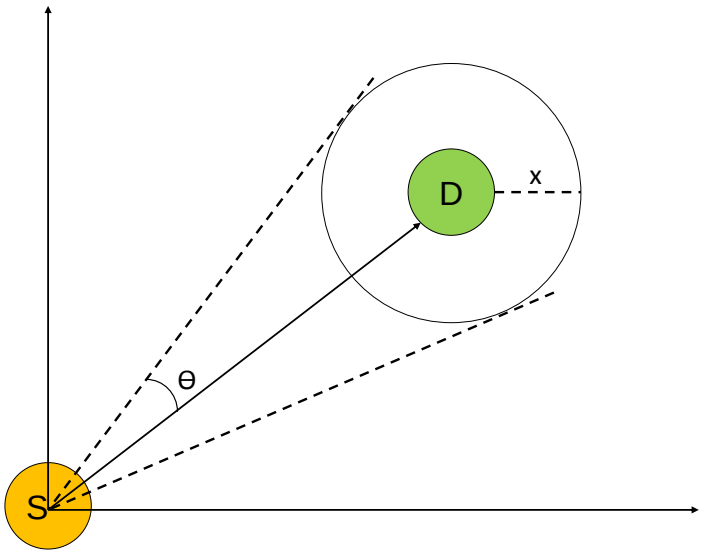

Figure 2.3. DREAM routing protocol(adapted from [3])

makes decisions on the frequency of control messages. Whenever a sender node wants to send data to a destination, it identifies the possible distance that can be covered by destination node over a period of time it would take for a packet to reach that destination. The sender node visualises that and creates a wedge covering a distance the destination node can travel. An assumption here is that the maximum velocity of destination is known before hand by the source. The sender node forwards data to its neighbours within the wedge. Neighbours do the same until the packet reaches its destination. In Figure 2.3, node S wants to send a data packet to node D. S knows location information of all its neighbours by exchanging control messages. S now identifies a wedge that encompasses the distance D might cover before the data packet reaches it. X represents the maximum distance D can travel.

LAR: Location-Aided Routing (LAR) [4] is an on-demand geographic protocol that uses the last known position and velocity information of the destination for sending route request messages. Based on the position and velocity information, the sender node calculates the *expected zone* and the *request zone*. Source S as shown in Figure 2.4 needs a route to destination D. S knows the location information of D at time t_0. The expected zone is the region S expects D would be located in at time t_1. S can identify this information based on the velocity at which D is moving. However, if S does not know the initial position of D, then the entire network region is considered as expected zone since D could be located anywhere in the network. The request zone as identified by source S is a region encompassing the expected zone and region covering the source and destination nodes. All nodes only within the request zone are required to flood the route-request initiated by S. The request zone can be expanded to cover a larger region of the network or even the entire region if the source S is not able to identify a route to destination D, as shown in Figure 2.4. This requires the intermediate nodes to figure out if they are present in the request zone or not, with two different schemes for the nodes to determine this. The first scheme consists of the sender sending a route request that contains the coordinates of a rectangle that contains the request zone. A node that receives this route request discards it if it is not within the rectangle and forwards if it is. Once the route request reaches the destination, it replies with the route reply message. The second schema does not explicitly define the request zone while sending the route request but instead forwards the packet based on the distance the sending node is from the destination, that is included in the route request.

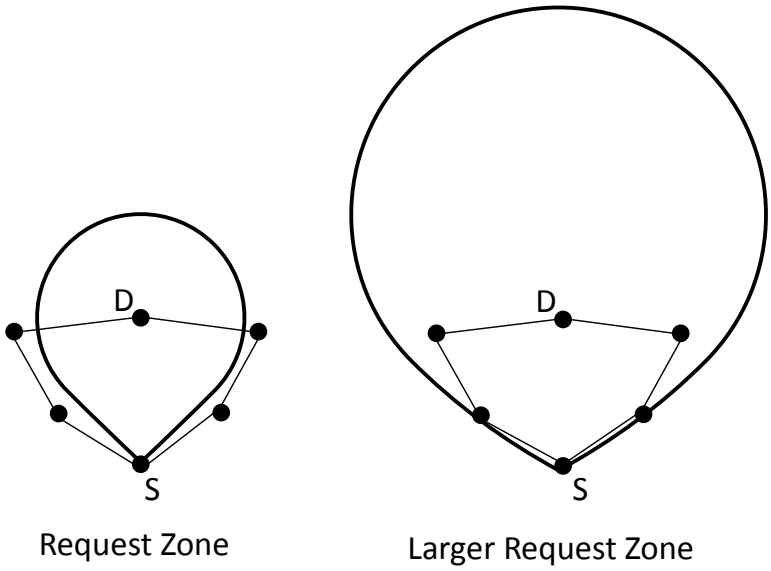

Figure 2.4. Location aided routing (adapted from [4])

2.1.2.3 Hierarchical Routing

Hierarchical routing introduces the concept of network hierarchies. The nodes in the network are divided a number of hierarchies specified by the routing scheme employed. Based on the hierarchies created, the tasks such as control message propagation or making routing decisions are given to nodes in an orderly fashion. In this section we will look at two hierarchical routing schemes: terminodes routing and grid routing.

Terminodes Routing: Terminodes routing proposes a two level hierarchy that includes both a proactive distance vector routing scheme and a greedy position-based scheme. Distance vector is used if sender node and receiver node are close to each other (considering the metric as hop count). The greedy position-based scheme is adopted if the sender node and the destination node are far away.

However, as the packet approaches the destination node, the scheme is changed back to distance vector. As we have seen earlier, greedy forwarding fails if the packet reaches a local maximum. To overcome this problem, the sender node includes a list of positions for the packet to traverse through in its header. The sender node gathers this position information from the nodes it was in contact with using the distance vector scheme. One disadvantage is that a sender node should continuously keep track of its neighbours movement, and for a highly dynamic environment, this involves a lot of overhead.

Grid Routing: Grid routing uses a similar hierarchy concept as terminodes routing. Proactive distance vector routing is used at the local level and greedy position-based scheme is used for long distance packet forwarding. A feature that grid routing introduces is proxy. Nodes that do not know their location information can also be part of the ad-hoc network. However, there should be at least one position-aware node within their reach using the proactive distance vector approach. This node acts as a proxy to the other position-unaware nodes within its reach. Packets destined for position-unaware nodes are forwarded to this position-aware proxy node, and using the distance vector approach these packets are then delivered to position-unaware nodes.

2.1.3 Advantages of position-based protocols

Topology-based routing protocols do not scale well for larger dynamic environments due to periodic broadcast of control messages. On-demand protocols generate route request queries for sending data to a new destination. They do so by using a flooding mechanism that leads to extra overhead in the network. On the other hand position-based routing protocols do not require any kind of main-

tenance for routing tables or route construction prior to the forwarding process. They can also forward data to valid next hop neighbours independently, taking into considerations the changes in topology, quality of service (QoS) related parameters such as delay or available bandwidth.

2.2 ANTP Protocol Suite

The Aeronautical network protocol (ANTP) suite is developed by the ResiliNets group at The University of Kansas for highly dynamic airborne networks. The aeronautical transport protocol (AeroTP), aeronautical network protocol (AeroNP), and aeronautical routing protocol (AeroRP) together form the ANTP suite.

2.2.1 AeroRP

AeroRP is a geographic routing protocol designed for highly dynamic airborne networks [12]. Contrary to the other MANET routing protocols that discover end-to-end paths, AeroRP makes only per-hop routing decisions. This is reasonable as the nodes in the airborne network move at very high velocities often leading to breakage of links after an end-to-end path is determined. AeroRP can operate in various modes based on the AN update mechanism, the mission requirements, and the presence of ground stations (GSAs) [11]. Based on the *AN update mechanism*, it can operate in either beacon or beaconless mode. In beacon mode, an AN advertises its presence by broadcasting periodic hello messages, whereas in beaconless mode no messages are sent out. Depending on the mission requirements, AeroRP can perform location-aware routing and location-unaware routing. In location-aware routing the GS and the ANs add node's geolocation information

to the control messages transmitted whereas in location-unaware routing they do not reveal the node's geolocation information. AeroRP can also operate in the presence of GSAs or without GSAs, discussed more in Chapter4.

2.2.1.1 Operational Aspects of AeroRP

The operation of AeroRP can be divided into two phases. The first phase of operation is the *neighbour discovery* phase. In this phase, an AN gathers as much information as it can about the network topology in the following ways:

- **Active snooping:** Active snooping is a mechanism in which the nodes snoop packets that are being exchanged among other nodes, extract the location information from them, and build or update their topology tables. To accomplish this, active-probing on the node's network interface must be enabled. Location information thus gathered is only valid for a time interval specified by neighborHoldTime. On expiration of this time-interval, the stored location information of a node is purged unless a new update with a higher expire time is received. This helps in keeping track of only the active neighbours in this highly dynamic environment.

- **Hello beacons:** Hello beacons are transmitted by the AN if it is not transmitting any data. This ensures that its neighbouring ANs are aware of the node's presence. These messages are usually broadcasted periodically over helloUpdateInterval with *time-to-live* (TTL) set to one hop.

- **Ground station advertisements (GSAs):** These are optional updates transmitted by the ground station during some missions that have a predetermined mission plan. These updates are broadcasted periodically and

are exchanged among all the ANs in the network.

The AeroRP modes explained earlier affect the various neighbour discovery processes. In beaconless mode the hello messages are not sent by any of the ANs. Therefore, neighbour discovery relies on *overhearing* the packets in the medium. Depending on the mission needs, if the AeroRP is operating in location-aware mode, then the ANs and the GS can use geolocation information in the hello messages and the GSAs. ANs can only be aware of their neighbours and the GS can only send out GSAs with topology information if AeroRP is operating in location-unaware mode.

The second phase of AeroRP operation is *data forwarding*. In this phase, the sender node determines the best next hop to forward a packet by using the topology table built in the *neighbour-discovery* phase. The *Time-to-intercept* (TTI) metric is used in determining this next hop neighbour [6,13,14]. TTI is calculated for every node from the topology table as:

$$\mathrm{TTI} = \frac{\Delta d - R}{s_{\mathrm{d}}}$$

where, Δd is the euclidean distance between the current location and a predicted location of a node based on the recorded location coordinates and velocity components, R is the common transmission range of all the nodes, and s_{d} is the recorded speed component. The assumption made here is that the nodes move at a constant speed during the interval for which we calculate the TTI value. TTI gives the estimate of time taken by the neighbours to reach within the transmission range of the destination. The neighbour with the lowest TTI value is chosen as the next hop neighbour and packets are forwarded to this neighbour.

2.2.2 AeroNP

AeroNP is an IP-compatible network protocol specifically designed for highly-dynamic airborne networks. Application systems and other devices on ANs are IP based. In addition to replicating the services provided by IP, AeroNP provides QoS (Quality of Service), congestion-control, and error-detection to the AeroTP transport layer [30]. QoS is provided by tagging packets based on priorities; AeroNP provides four levels of priorities. The AeroRP control packets are always given the highest priority. Mission specific command and control data is given higher priority compared to application data. The packets tagged with a particular priority are queued in specific buffer classes. The congestion-control mechanism is accomplished by implementing a cross-layering mechanism with the iNET TDMA MAC layer [31]. The Congestion indicator (CI) field in AeroNP specifies the congestion level at a node. If a node identifies that the MAC buffer is full, it increases its congestion indicator value in the AeroNP header. Neighbouring nodes do not forward packets to a node with high CI value, unless it is the destination. The wireless medium is error-prone leading to packet corruption, and detecting these errors at the destination and waiting for the source to resend the packet increases the end-to-end delay. The AeroNP corruption indicator and HEC-CRC (header error check – cyclic redundancy code) fields provide the error detection. Depending on the mission requirements, geolocation information can be included in the AeroNP header. We designate the AeroNP header with the geolocation information as the *extended header*, whereas the AeroNP header without the geolocation information as is referred to as the *basic header*.

Chapter 3

Implementation of ns-3 Models

Simulation has been the backbone of MANET research [8, 32], since the simulation environment provides easily accessible resources to study new protocols and models. The ns-2 simulator [33] has been widely used due to its open-source model which is appropriate for the academic research community. In response to a number of its deficiencies, the ns-3 discrete event network simulator [34] is under development, providing greater flexibility, modularity using C++, evolvability, and support for heterogeneity including hybrid wired and wireless models.

Despite its advantages, ns-3 is relatively new with few protocol models yet incorporated into its release distribution [35]. As part of our contribution from the ResiliNets group we have modelled DSDV routing protocol[1], 3D–Gauss-Markov mobility model[1], TDMA MAC protocol[2], and DSR routing protocol[2]. In this chapter we present the implementation details of DSDV routing protocol and TDMA MAC protocol. Section 3.1 presents a detailed explanation of DSDV's headers, its routing table, transmitting and processing DSDV advertisements,

[1] incorporated in the mainline release of ns-3
[2] currently under testing

and data packet buffering. Section 3.2 details the implementation aspects of a centralised TDMA controller and TDMA frame transmission and processing. It also highlights the differences in operation of a TDMA MAC protocol compared to the 802.11 MAC protocol.

3.1 Implementation of DSDV in ns-3

The MANET protocols in early releases of ns-3 were limited to just the optimised link state routing (OLSR) and the ad hoc on-demand distance vector (AODV) protocols. Thus we have developed an ns-3 implementation of the destination-sequenced distance vector (DSDV)[1] routing protocol. DSDV is one of the earliest MANET routing protocols proposed [20] and provides a baseline for performance comparisons against AeroRP protocol with ground station updates.

3.1.1 DSDV module for ns-3

This section describes our implementation of DSDV, which became part of ns-3 since ns-3.10 release. The main components of the DSDV implementation are routing update mechanisms, route table creation, and route maintenance. DSDV maintains valid routes and flushes out invalid routes based on the periodic update interval. We implemented an optional packet buffering mechanism that was not part of the initial DSDV design [20]. This feature is implemented for testing the performance of the protocol with and without packet buffering and also to provide users with more options. All the attributes used in this implementation are listed in Table 3.1. The relation between all the classes implemented in this module are

[1]This section is mainly based on the "Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3" paper [36], for which the author of this thesis is primary author

shown in Figure 3.1. We implemented the DSDV routing protocol `ns3::dsdv::RoutingProtocol` in ns-3 by extending from the abstract base class `ns3::Ipv4RoutingProtocol`. The `ns3::dsdv::DsdvHeader` is extended from `ns3::Header`. We have also declared `ns3::dsdv::RoutingTableEntry` to store the updates of a node and `ns3::dsdv::RoutingTable` to store all these entries in a table. Similarly we have declared the `ns3::dsdv::QueueEntry` class to store a packet and `ns3::dsdv::RequestQueue` to store all the queued entries. The main class that glues all these together is the `ns3::dsdv::RoutingProtocol` class. An in-depth explanation of all these classes is presented in the following sections.

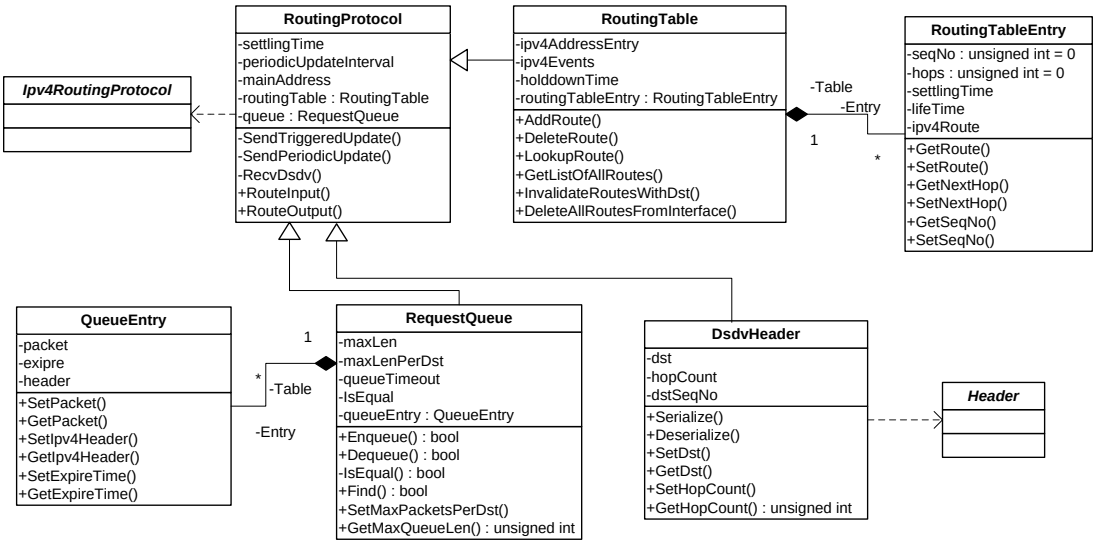

Figure 3.1. DSDV class diagram

Attribute	Defaults	Summary
PeriodicUpdateInterval	15 s	Time interval between exchange of full routing tables among nodes
EnableWST	true	Enables Weighted Settling Time for the updates before advertisement
SettlingTime	6 s	Minimum time duration an update is stored before transmission
WeightedFactor	0.875	Weighted factor for the settling time if EnableWST is true
EnableBuffering	true	Enables buffering of data packets if no route to destination is available
MaxQueueLen	100	Maximum number of packets that can be queued
MaxQueueTime	30 s	Maximum time duration for which packets can be queued
MaxQueuedPacketsPerDst	5	Maximum number of packets that can be buffered per destination
Holdtimes	3	Number of times PeriodicUpdateInterval to purge a route
EnableRouteAggregation	false	Enables aggregation of DSDV updates over a period of time
RouteAggregationTime	1 s	Time over which DSDV updates are aggregated

Table 3.1. DSDV attributes and default values

3.1.2 Header

The DSDV message header (`DsdvHeader`) is 32 bits wide with the total header size of 12 bytes as shown in Figure 3.2. The fields in the DSDV header are the node's IP address, the number of hops required to reach that node, and its last known sequence number. The latter two are 32-bits long in our implementation to provide word alignment and allow simulation of very large networks, even though the ns-2 implementation used 16-bit fields. Note that unlike AODV and OLSR, there is no DSDV RFC to guide standards compliance. DSDV is encapsulated in User Datagram Protocol (UDP) segments that are then encapsulated in IP packets, as shown in Figure 3.3.

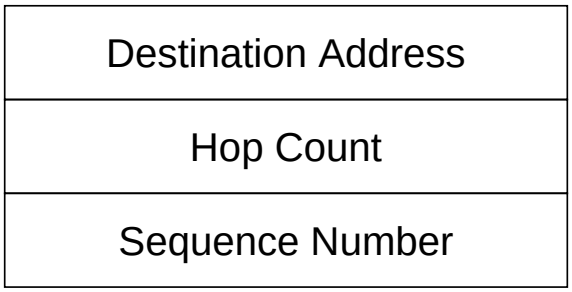

Figure 3.2. DSDV message header

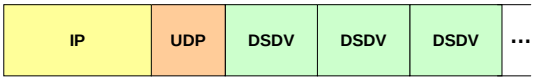

Figure 3.3. DSDV header encapsulation

3.1.3 Routing Table

The structure of the DSDV `RoutingTable` is implemented as follows. Each entry implemented by the `RoutingTableEntry` class corresponds to a node in the network and the entry is mapped to that node's IP address. Every entry stores

the following attributes of a node: its IP address, interface address, a pointer to its ns-3 net device, last known sequence number of the node, hop-count to reach the node, timestamp of the last update received for the node, and the settling time for that node. Also, we maintain a boolean value that specifies whether the entry for this node has changed since the last periodic update. This helps filter DSDV updates that are broadcasted through the trigger update mechanism. The `RoutingTable` class has methods to add, delete, update, look up, and print entries. It also defines the event functions explained in Section 3.1.5.2.

DSDV maintains two routing tables: a permanent routing table and an advertising routing table. These tables store the permanent stable routes and the recently received routes respectively. The recently received routes might be unstable; therefore, when the node identifies a route to be stable, it advertises that route and moves it to the permanent routing table. This mechanism of identifying stable routes is done using *SettlingTime*, explained in detail in Section 3.1.4. Furthermore, a node can identify the stability of a route based on the sequence number and hop count received through the update, explained in Section 3.1.5.

3.1.4 Routing Advertisements

A node combines all the DSDV messages that it has to transmit into a single packet over `RouteAggregationTime`, if `RouteAggregation` is enabled. However, to keep the packet size under the maximum transmission unit (MTU) in the implementation, we split the packets and send them separately if the packet size is longer than the MTU. As mentioned earlier, DSDV sends both periodic update messages and trigger messages. As soon as the routing protocol in the node is initialised, the node broadcasts its DSDV update message to the network to

announce its presence. Each node periodically broadcasts its own routing table and all the nodes that are in range of this advertisement use this information to update their routing tables. They may further trigger these updates to other nodes in their broadcast range. This mechanism is also used to keep the neighborhood relationship alive. One of the attributes that can be set for the routing protocol is the duration between these periodic updates, known as *periodic update interval*, using `PeriodicUpdateInterval`. It specifies the time duration for which a node has to wait before broadcasting its routing table. A node uses the trigger update mechanism when there are only a few updates to be transmitted. However, if the node identifies that the number of updates sent per trigger is comparable to that of a periodic update, then it sends a periodic update instead.

One more feature of DSDV routing protocol is the *settling time*, which is used to prevent the advertisement of an unstable route that arrives at the node before a stable route. Since DSDV uses broadcasting to propagate these changes, it would create unnecessary overhead in the network. Thus, a node waits for the period of `SettlingTime` before propagating any update. However to make sure updates for stable routes are not delayed, we use the attribute `WeightedFactor`. This is used to calculate the weighted average of the settling times for the updates received from a node. If the update is for an old and stable route, the settling time decreases. A node can not process multiple update messages simultaneously. If the nodes are highly mobile, the node might have to send many updates as there would be a lot of route changes. This leads to more overhead in the network that may increase the number of collisions. To reduce overhead, we use `RouteAggregation`. This optional feature enables multiple update messages to be sent out as a single update message. The period over which routes are aggregated can be modified by

`RouteAggregationTime` attribute.

3.1.5 Processing of Updates

As mentioned in Section 3.1.4, a node might receive many updates stacked within a single packet. Since the DSDV header size is fixed to 12 bytes, we iterate over the packet until it is empty to extract all the 12 B DSDV control messages. Each message is processed as it is extracted. We first verify the destination address in the extracted message. If it is same as the node's IP address, the message is discarded. If not, the protocol verifies whether the received update is for a new route with a valid sequence number. In this case the route is added to the permanent routing table and broadcast immediately. Otherwise, if the node already has an update for that IP address the protocol verifies the sequence number. If the sequence number is odd and if the node from which this update was received is the next hop neighbour in the table, then the route is deleted from the routing table and triggers an update of this broken route to other nodes immediately. However if the sequence number is valid we have three cases in which the sequence number can relate to the sequence number from the table:

- *Received* > *Local*: The protocol verifies the received hop-count with the local value of hop count. If they are not equal, the node updates its local entries in the advertising routing table and waits for settling time period if `SettlingTime` is enabled. This is implemented using events in ns-3. This mechanism is explained briefly in Section 3.1.5.2. If the received hop-count is same as the local value, then the node does not wait for the settling time interval as this is an update for the stable route.

- *Received* = *Local*: If the received hop-count is less than the local value,

then the local value is updated and the protocol waits for settling time to make sure that this update is not an unstable one. If it does not receive any further update for that destination address, the protocol updates the permanent table with this update and triggers this update back to all its neighbours. However if the received hop-count is greater than or equal to the local value, the message is discarded.

- *Received* < *Local*: The protocol discards this update message as it already has a most recent update from that destination.

After processing messages from the packet, the `SendTriggeredUpdate` method is called. `SendTriggeredUpdate` iterates over the advertising routing table, computes all the needed updates, and creates a new packet with these updates and broadcasts.

3.1.5.1 Stale Entries

DSDV has a mechanism of removing stale entries from the node's routing table. If a node does not receive any updates for a destination over a period of time, it removes that entry from the routing table. In our implementation, DSDV waits for `Holdtimes` × `PeriodicUpdateInterval` interval. The default value of `Holdtimes` is set to 3, i.e. a node waits for 3 times the `PeriodicUpdateInterval` before deleting the route. Furthermore, the node must delete all the routes for which the deleted neighbour was the next hop.

3.1.5.2 Event Processing

In the implementation of DSDV we use EventId `ns3::EventId` to schedule events and keep track of them. These are declared in the `RoutingTable` class.

The IP address of a node is mapped to the event id. We use these events to keep track of the updates in advertising table and broadcast them when their settling time is complete. When a node receives an update for a destination that is already waiting in the advertising table, the running event might be replaced by a new one depending on the new update received.

3.1.6 Packet Buffering

We have implemented a buffering mechanism for DSDV although it is not part of the DSDV as originally described [20], to allow fairer comparisons with *disruption-tolerant networks* (DTN) and *domain-specific* MANET routing protocols that do buffer packets that cannot be immediately sent [13, 37]. We implemented two classes in a manner similar to the routing table implementation. `QueueEntry` class is the entry that is stored in the queue, implemented from `RequestQueue` class. If the destination address for a packet is not present in the protocol's routing table, then the packet is buffered. As DSDV is a proactive protocol, it does not initiate any route discovery mechanism to identify the route to that destination. It has to only rely on the messages received from its neighbours through trigger and periodic updates. DSDV periodically verifies the buffer and look for packets with valid routes in the routing table and transmits them. By default, our DSDV implementation buffers up to 5 packets per destination. This can however be changed by modifying the `MaxQueuedPacketsPerDst` attribute. Furthermore, packets that are buffered for a long time are dropped from the queue. The time interval for which a packet can be buffered is set using `MaxQueueTime`. By default packet buffering is enabled, but this can be disabled by setting `EnableBuffering` to `false`.

3.1.7 Parameter Tuning

An advantage of DSDV is that it is relatively simple compared to other MANET routing protocols. It is also similar to the conventional wired distance-vector routing protocols, with only minimum adaptations made. However, the drawback of DSDV is that its periodic overhead for broadcasting is unavoidable even if the network is static. If the node density increases in the network, the routing table also becomes larger. This leads to more updates with larger packet sizes. With a highly dynamic network, the routing updates may take up the available bandwidth of channel. Furthermore, before the time of update, intermediate nodes may use stale information to forward packets. Thus proper choice of `PeriodicUpdateInterval` and `SettlingTime` is important in a highly mobile environment.

3.1.8 DSDV Module Evaluation

To evaluate the performance of our DSDV routing protocol implementation, we performed simulations using the ns-3.9 version of the network simulator[2]. To verify its functionality, we investigate the DSDV performance with varying node densities as well as compared to the other existing MANET routing protocols in ns-3: OLSR and AODV. Note that a comparison to the DSR implementation currently in progress is future work.

3.1.8.1 Performance Metrics

The performance metrics for evaluation of the DSDV routing protocol are packet delivery ratio (PDR), routing overhead, and delay.

[2]Before our DSDV was included in the ns-3 distribution in ns-3.10.

- **Packet Delivery Ratio PDR**: The number of packets received divided by the number of packets sent by the application.
- **Routing Overhead**: The fraction of bytes used by the protocol for DSDV control messages
- **Delay**: The time taken by the packet to reach the destination node's MAC protocol from the source node's MAC protocol.

3.1.8.2 Simulation Setup

We performed the simulations over an area of 1500 $\times$ 300 m^2. All the simulations were averaged over 10 runs with each simulation running for 1000 s. Simulations were performed with varying node densities: 10, 20 and 30 nodes. The communication model is peer-to-peer communication with as many flows as the number of nodes in the network. We initially performed some simulations with 1000 byte packets but observed that the PDR was low, therefore we used a packet size of 64 bytes based on previous study [38]. All the nodes are configured to send 4 packets/s. Using this lower packet size, we can correctly evaluate the performance of the protocol. We use the ns-3 `On-Off` application to generate CBR (constant-bit rate) traffic. The 802.11b MAC is used over Friis propagation loss model to limit the transmission ranges of nodes. The transmit power was set to 8.9048 dBm to achieve a 250 m transmission range. The mobility model used is random waypoint with random velocities from 0 – 20 m/s and pause times of 100 – 800 s. When comparing DSDV performance against AODV and OLSR, we use 0 s pause time. DSDV performance with optional buffer mode enabled was analysed. We used the default DSDV parameters values described above except for `PeriodicUpdateInterval` which was varied among {4, 5, 8, 12, 15, 30} s

and `SettlingTime` which was varied among {0, 1, 2, 3, 4, 5, 6} s. Some of the simulation parameters were chosen based on the previous MANET comparison studies [38].

3.1.8.3 Simulation Analysis

In the first scenario, we vary the pause time in the random waypoint mobility model so that we can analyse the performance of DSDV in both mobile and static scenarios. For this scenario, the `PeriodicUpdateInterval` is set to 15 s and `SettlingTime` is 6 s. Figure 3.4 shows the variation of PDR by varying the pause times.

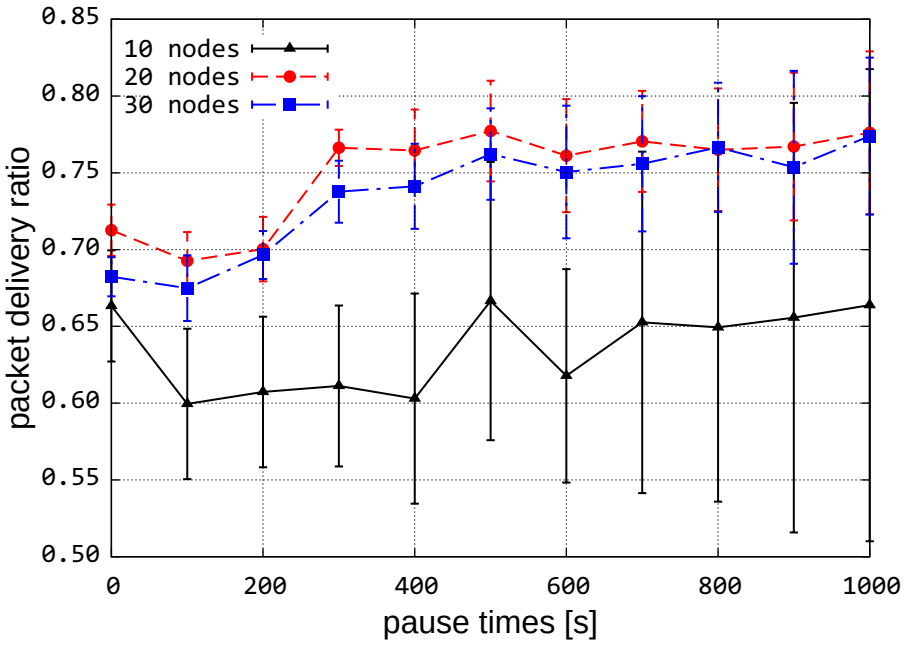

Figure 3.4. PDR with varying pause time

We can see that as the number of nodes is increased the packet delivery ratio also increased. This is due to the fact that when there are only 10 nodes, the chances of link breaks and network partitioning is more likely to happen than when

there are more nodes making the network connected for most of the time. However, PDR for 20 nodes is greater than that for 30 nodes for all pause times. This might be because as the node density increases, the routing overhead also increases and this leads to more collisions in the network. Note the 95% confidence-interval error bars in Figure 4. As the pause time increases, so does the variation in packet delay (as depicted by error bars) for all the 3 curves for 10 nodes, 20 nodes and 30 nodes. This can be attributed to how the nodes were positioned in the network initially since very long pause times reduce movement from the initial position.

The routing overhead for different node densities with varying pause times is shown in Figure 3.5. This plot shows that overhead increases with the number of nodes. This is expected for DSDV since it is a proactive protocol and every node keeps track of all the other nodes in the network; when a node sends out a periodic update, it is flooded to all other nodes. Depending on the changes based on an update received, a node may further trigger updates to other nodes.

The overhead as shown in Figure 3.5 slightly increased for all the 3 curves moving from a pause time of 0 s to 100 s. With zero pause time the nodes collect less information from the network because they are continuously moving. With the larger pause time of 100 s they collect more information from the network. This translates to more updates. Furthermore, as pause time is increased, the overhead is reduced.

We also consider the packet delay for data packets between source and destination. Figure 3.6 shows the variations in packet delay (as depicted by error bars) increase as the pause time is increased for all the 3 curves for 10 nodes, 20 nodes and 30 nodes. This is because as the pause time is increased the nodes are immobile for longer durations and thus the link connectivity depends on the

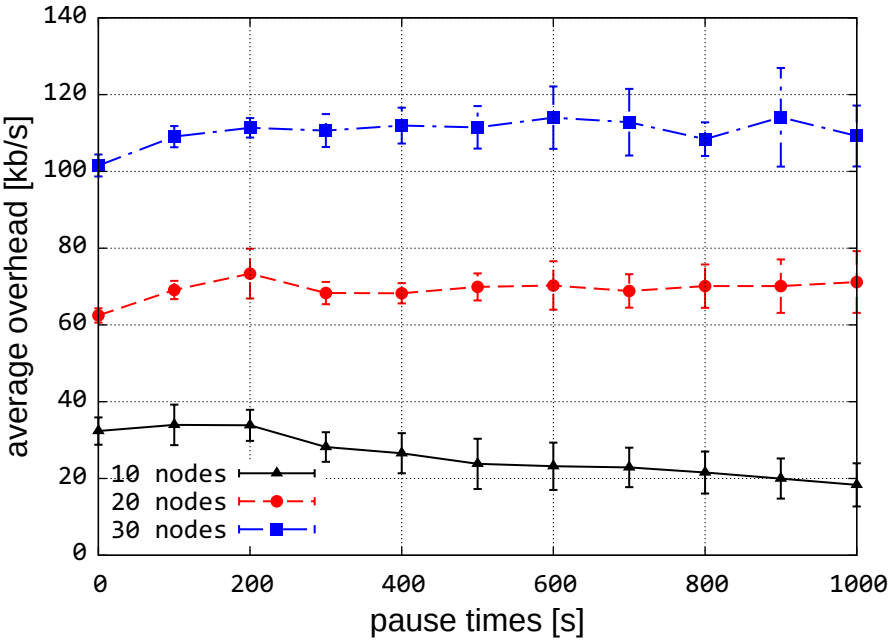

Figure 3.5. Overhead with varying pause time

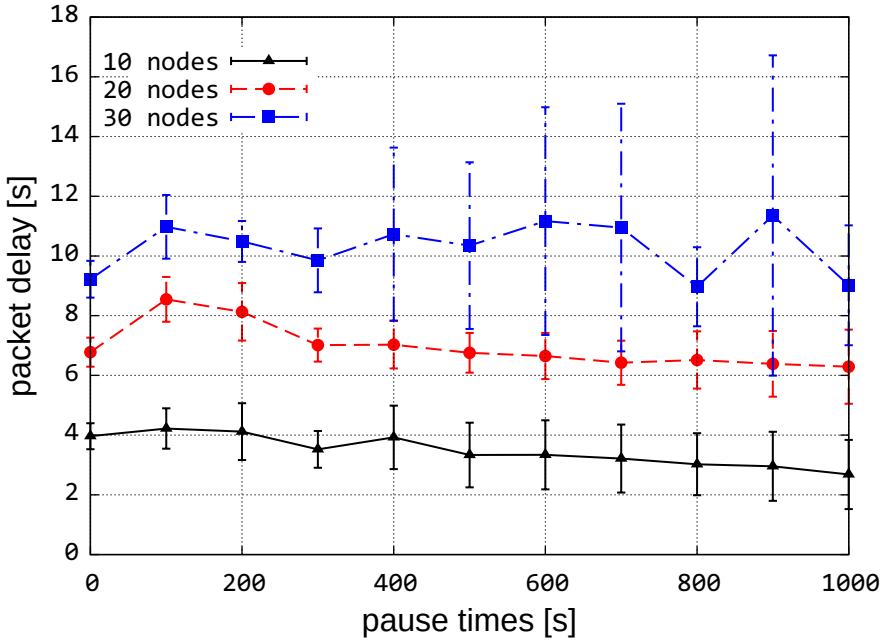

Figure 3.6. Packet delay with varying pause time

position of the nodes, which directly affects the packet delay.

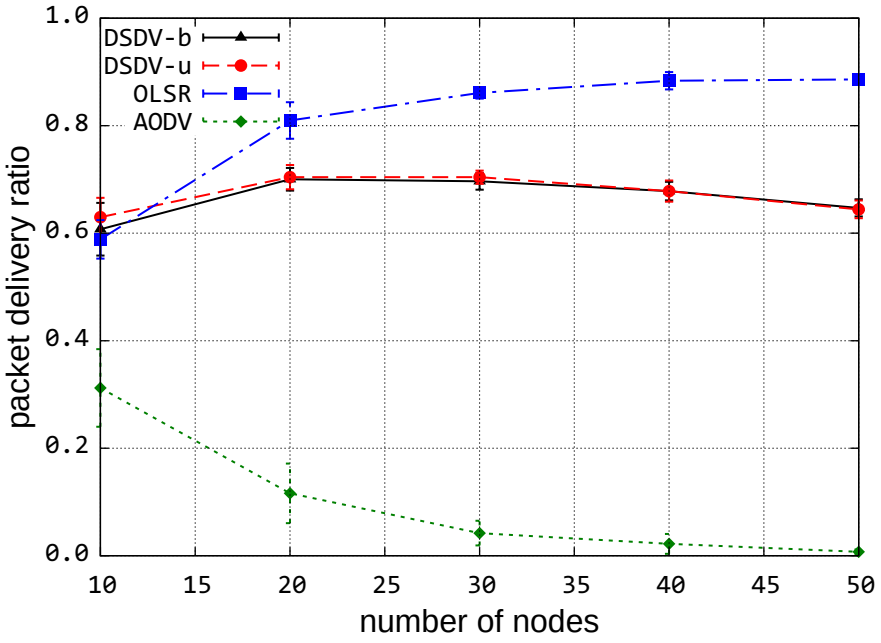

Figure 3.7. PDR with varying node density

In Figure 3.7 we compare the packet delivery ratio of existing MANET routing protocols implemented in ns-3 with DSDV. From the plot we can clearly see that OLSR outperforms DSDV-buffer mode, DSDV-unbuffer mode, and AODV. This is expected as OLSR implementation in ns-3 exchanges TC messages every 5 s [21], thus the routing tables are computed/re-computed every 5 s. However DSDV uses a `PeriodicUpdateInterval` of 15 s making the convergence of nodes running OLSR quicker compared to those running DSDV. In DSDV the routes are not always accurate as it depends only on periodic and trigger messages to update the routes. AODV's performance was expected to be higher, however the current implementation of AODV has some bugs that need to be fixed[3].

[3]We have been working with ns-3 developers to report AODV performance issues, and the situation has been improving.

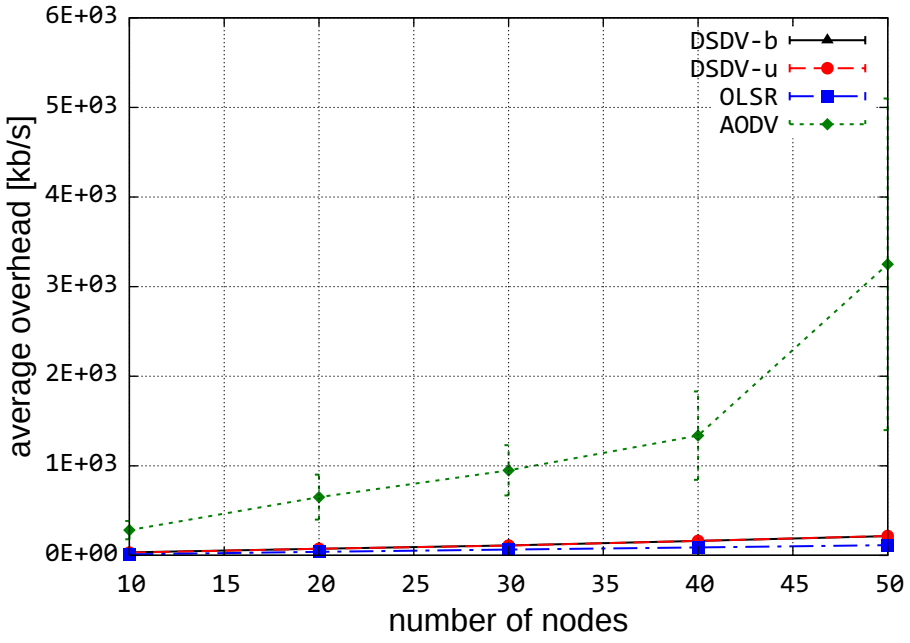

Figure 3.8. Overhead with varying node density

In our analysis, we also compare the routing overhead involved with all these protocols, AODV incurs significant overhead shown in Figure 3.8. DSDV and OLSR generates about 112 kb/s and 65 kb/s of routing overhead respectively for 30 nodes. However as the number of nodes increases, the overhead increases as well. For a 50 node simulation, DSDV incurred an overhead of 215 kb/s compared to 113 kb/s for OLSR.

We analyse the packet delay for these protocols. The packet delay is greater for DSDV when compared with OLSR as shown in Figure 3.9. For a 30 node simulation, packet delay for DSDV was 10 s whereas it was 6 s for OLSR. Since these scenarios were generally connected, the results for DSDV-buffer and DSDV-unbuffer mode results were not significantly different. The performance of the ns-3 AODV model is considerably less than expected.

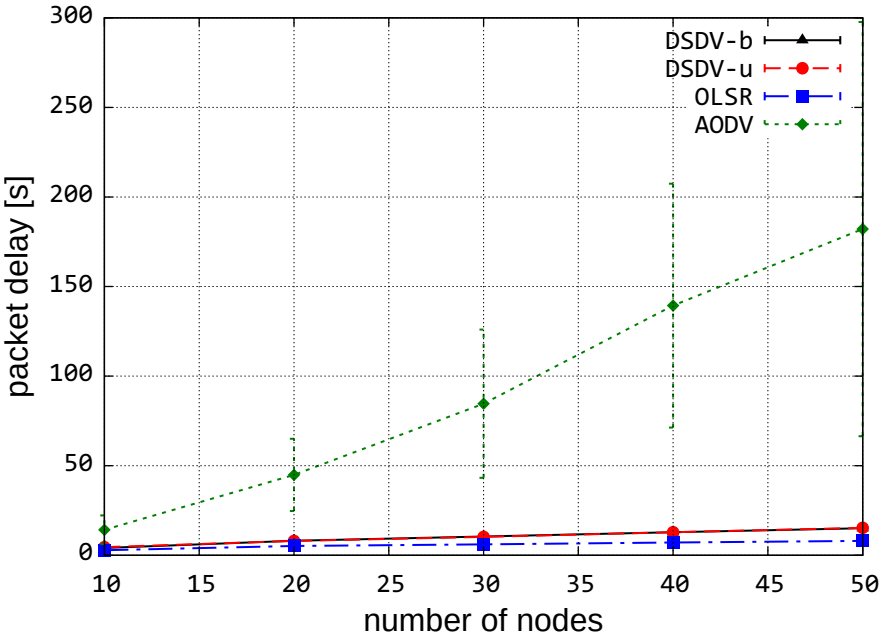

Figure 3.9. Packet delay with varying node density

3.2 Time Division Multiple Access

Time division multiple access is a contention-free medium access protocol. The channel bandwidth that is shared by all nodes in the network is partitioned into time slots for dedicated use among those nodes. Each node transmits data only during its allotted time slot. The transmission slots are usually of fixed time intervals. In an airborne telemetry network, the GS takes responsibility of assigning transmission slots to the AN's. The GS splits the frame into slots depending on the number of AN's and the number of slots it requires for itself. Each transmission slot is separated by a guard interval so that the transmissions do not overlap. The value of the guard interval is determined by the GS. It is usually the amount of time it takes for a packet to travel the distance specified by the transmission range. In this simple TDMA model, it is assumed that the

clocks of the nodes are synchronised.

3.2.1 Difference between IEEE 802.11 and TDMA

Many collisions in MANETs are caused by hidden node problem. When two nodes which are outside the transmission range of each other send data to the same receiver, packets from both transmitters collide at receiver. This is known as hidden node problem. The protocols with proposed solutions can be divided into two categories, contention-based and contention-free. IEEE 802.11 is a contention based MAC protocol whereas TDMA is contention-free MAC protocol.

3.2.2 Implementation of TDMA protocol in ns-3

The ns-3 implementation of TDMA[4] uses a centralised TDMA controller that assigns transmission slots to various nodes in the network. Figure 3.10 shows the class-interaction diagram between various classes used in the implementation. `ns3::TdmaController`, `ns3::TdmaCentralMac`, and `ns3:TdmaMacQueue` are the major classes in this implementation. `ns3::TdmaController` controls the scheduling aspect of the protocol. TDMA frame processing, creating MAC headers and trailers, and MAC callback mechanisms are handled by `ns3::TdmaCentralMac`. `ns3::TdmaMacQueue` takes care of the packet queueing and dequeuing.

3.2.2.1 `ns3::TdmaCentralMac`

`ns3::TdmaMac` is the base class from which `ns3::TdmaCentralMac` is derived. The current implementation considers a simple centralised TDMA MAC. However, considering the other possible implementations of distributed TDMA models, we

[4]The ns-3 TDMA model was developed as part of the thesis; the 802.11 model is in the standard ns-3 distribution

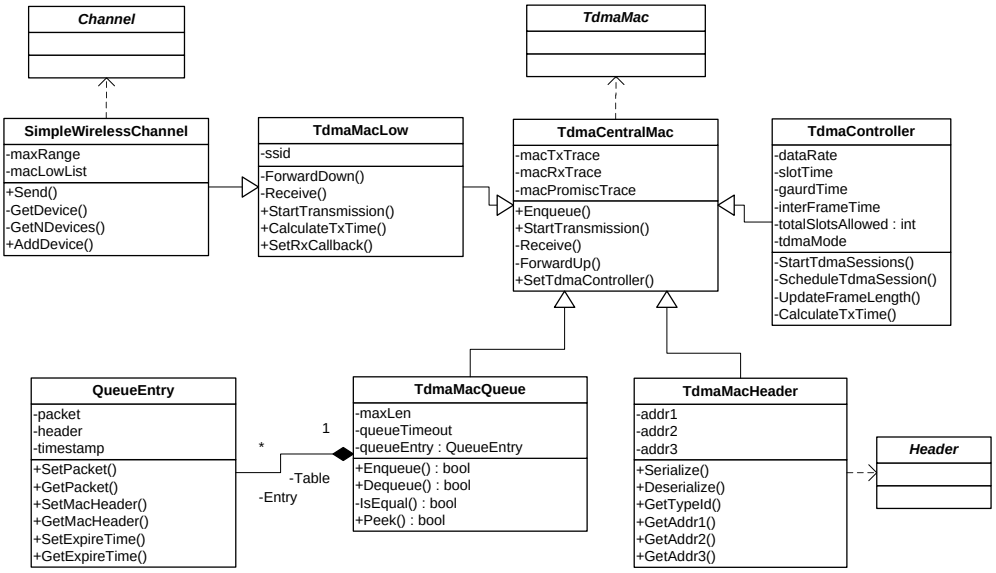

Figure 3.10. TDMA class diagram

created a common base class so that other implementations could be derived from it. All the data packets ready for transmission by the node are sent down either from AeroNP or IP to TdmaCentralMac. TdmaCentralMac upon receiving the packets, enqueues them and waits for its turn to transmit. As soon as this node gets its turn to transmit, TdmaCentralMac looks up the `ns3::TdmaMacQueue` for any queued packets. It then iteratively dequeues packets from the queue, attaches the MAC headers and trailers and them sends them to `ns3::SimpleWirelessChannel`. Before sending them, TdmaCentralMac calculates the transmission time required based on the packet size and data rate. It adds up the transmission times of all the packets sent and compares it with the transmission time-slot allotted to it by the TdmaController. If it could not transmit any more packets in that slot, the loop terminates stopping further transmissions. SimpleWirelessChannel forwards

the packets to all the nodes which are within the MaxRange attribute value specified by the user at the start of simulation. TdmaCentralMac also takes care of the packets received from SimpleWirelessChannel. It removes the attached MAC headers and trailers and forwards the packet to either AeroNP or IP.

3.2.2.2 ns3::TdmaMacQueue

TDMA maintains a drop-tail queue to store packets received from the network layer until it gets its transmission slot. The attributes that can me modified for this class are MacQueueLength and MacQueueTime. So all the packets trying to be enqueued after the queue size reaches MacQueueLength are be dropped and packets stored in the queue for a time-interval longer than MacQueueTime are also dropped.

3.2.2.3 ns3::TdmaController

ns3::TdmaController takes care of all the scheduling aspects of the protocol. It initiates the TDMA sessions and authorises the nodes to transmit in the slots specified by it. The number of slots alloted for transmission along with the slots durations are provided to it as attributes specified by the user at start of simulation. The list of attributes along with their default values associated with TdmaController are shown in Table 3.2. A ns3::TdmaHelper takes all these attributes along with a list on nodes and initialises the TdmaController. TdmaController maintains a list of MAC pointers associated with all the nodes. Based on the slot assignment provided by the user, this list is populated by TdmaHelper class before the simulation starts. The user can provide the slot assignments for nodes either through the simulation script or an external file. After the simula-

tion starts, the TdmaController initiates scheduling of TDMA sessions based on the node ids. It calls the `ns3::TdmaCentralMac` from its list of MAC pointers and instructs the node that it could transmit for a particular SlotTime. As soon as the transmission slot for that node is complete, the TdmaController waits for GaurdTime and then calls the next node from the list and so on. Once all the nodes from the list are assigned a transmission slot, the controller waits for InterFrameTime before starting with the same procedure again.

Attribute	**Default Value**	**Summary**
DataRate	11 mb/s	The default data rate for links
SlotTime	1100 μ s	The duration of a transmission slot
GuardTime	100 μ s	Guard time between transmission slots
InterFrameTime	10 μ s	The wait time between consecutive frames
TotalSlotsAllowed	1	Number of total slots allowed per frame

Table 3.2. Attributes with default values for TdmaController

Chapter 4

Design of AeroRP with GS

The main goal of this thesis is to design and implement the GSAs (ground station advertisements) as one of the neighbour discovery processes in AeroRP. In this chapter we will look at how the GSAs are broadcasted by the GS and how they are processed by the ANs[1]. This chapter is organised as follows. Section 4.1 details the message formats used by AeroRP to send GSAs. The mechanism employed by the GS to broadcast these GSAs is briefly explained in Section 4.2 and processing of GSAs by the ANs is explained in Section 4.3.

4.1 AeroRP Header Format

In this section we will look at the AeroRP message header formats. AeroRP uses TypeHeader, GSGeoLocationHeader, and GSTopologyHeader. The latter two are exclusively used by the GS to send our GSAs.

[1]This chapter is mainly based on the "Performance Analysis of AeroRP with Ground Station Updates in Highly-Dynamic Airborne Telemetry Networks" paper [39], for which the author of this thesis is primary author

4.1.1 Type Header

The introduction of ground station updates required a change in header format used in [6]. For compatibility reasons after introducing AeroNP, AeroRP message formats were significantly modified so that a common TypeHeader message can be attached at beginning of every AeroRP update. TypeHeader shown in Figure 4.1 specifies the type of AeroRP message attached to it, flags required for processing the attached AeroRP message(s), and header length. A summary of the contents of TypeHeader follows:

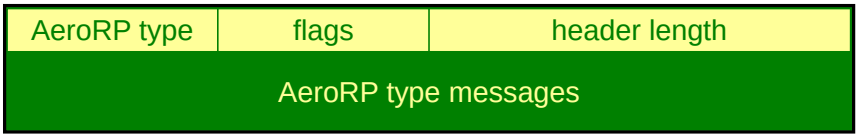

Figure 4.1. Packet format for TypeHeader

- AeroRP type: 8 bits

  The AeroRP type field indicates the type of AeroRP message attached below. The type can be HelloHeader, GSGeoLocationHeader, and GSTopologyHeader.

- flags: 8 bits

  The eight bits of flags field are used to unpack and process attached AeroRP message.

  - bit 0: Reserved, must be set to 0
  - bit 1: Reserved, must be set to 0
  - bit 2: Reserved, must be set to 0
  - bit 3: Reserved, must be set to 0

- bit 4: This bit helps a node to identify a packet with GS updates sent by neighbouring node. This message is initiated by a neighbour node if it discovers that this node has outdated information from the GS.
  0 = No GS update attached, 1 = GS update attached.
- bit 5: (GF) This bit is used by the GS to differentiate between periodic updates and trigger updates.
  0 = Trigger update, 1 = Periodic update.
- bit 6: (GE) This bit is used along with HelloHeader. It tells if a node wants to exchange ground station updates with other nodes in network.
  0 = Disable GS sequence number exchange, 1 = Enable GS sequence number exchange.
- bit 7: (EB) This bit tells the node whether to rebroadcast this update message again or not.
  0 = Disable rebroadcast, 1 = Enable rebroadcast.

- header length: 8 bits

  This field specifies the total AeroRP message header length attached to the packet.

- AeroRP type message: variable bits

  This field is a placeholder for the other AeroRP messages to be attached as specified in the AeroRP type field.

4.1.2 Hello Header

HelloHeader format is significantly modified since [6]. The required HelloHeader fields such as node's geolocation and velocity information fields are moved to the

AeroNPHeader that is explained in Chapter 5. So if a node wants to send a hello message, it uses a TypeHeader with AeroRP type as hello and attaches an AeroNPHeader to it.

4.1.3 GSGeoLocation Header

GSGeoLocation is introduced to broadcast geolocation information of all the ANs. It carries a node's geolocation coordinates, velocity components, start time, and end time. Figure 4.2 illustrates the fields used and a detailed explanation follows.

node id	reserved
x-coordinate	x-velocity
y-coordinate	y-velocity
z-coordinate	z-velocity
start time	end time

Figure 4.2. Packet format for GSGeoLocationHeader

- node id: 16 bits

  The node id field indicates the node's 16-bit id generated by AeroGateway, whose geolocation information and velocity components are present in the fields below.

- reserved: 16 bits

  This field is currently reserved for future use.

- x-coordinate: 19 bits

  This field specifies the cartesian x coordinate based on the current location of the node.

- x-velocity: 13 bits

  This field specifies the velocity component of the node in x direction.

- y-coordinate: 19 bits

  This field specifies the cartesian y coordinate based on the current location of the node.

- y-velocity: 13 bits

  This field specifies the velocity component of the node in y direction.

- z-coordinate: 19 bits

  This field specifies the cartesian z coordinate based on the current location of the node.

- z-velocity: 13 bits

  This field specifies the velocity component of the node in z direction.

- start time: 16 bits

  start time is used if an AN follows a pre-determined trajectory. This information is used by the other ANs to start using the location information of this AN from start time in making routing decisions.

- end time: 16 bits

  end time is used if the GS identifies that this AN will deviate from a predetermined path at a particular time. This information is used by the other ANs to stop using the location information of this AN after end time in making routing decisions.

4.1.4 GSTopology Header

GSTopologyHeader is used to broadcast the link information of all the nodes. The fields present in the GSTopologyHeader are 16-bit node-ids of both the nodes forming this link, the link start and expire times, and the link cost. Figure 4.3 illustrates the fields followed by a detailed explanation.

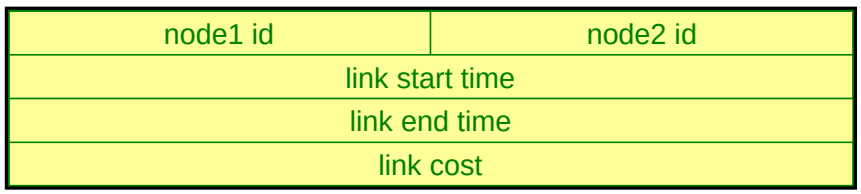

Figure 4.3. Packet format for GSTopologyHeader

- node1 id: 16 bits

  The node1 id field indicates a node's 16-bit id generated by AeroGateway that formed a link with node whose id is present in node2 id field.

- node2 id: 16 bits

  This field specifies a node's 16-bit id that formed a link with the node whose id is present in node1 id field.

- start time: 16 bits

  This start time specifies the time at which this link is formed.

- end time: 16 bits

  This end time specifies the time at which this link is predicted to go down.

- link cost: 32 bits

  Link cost is used by the AN to identify a shortest path to a destination. The GS can take many factors in determining this link cost. The lower

the link cost, the better it is to send traffic over it. Some of the factors in determining link cost are highlighted below.

1. Duration for which a link will be active.
2. Links where one of the node has more resources or has more paths to a destination

4.2 Operations of Ground Station

The ground station is responsible for monitoring location information of all the ANs and broadcasting updates on their location to the other ANs.

4.2.1 GS Update Mechanism

A ground station broadcasts updates for all the ANs periodically over a time-interval called the periodicUpdateInterval. Depending on velocities of the ANs and the frequency at which they change direction, the periodicUpdateInterval is set. The frequency at which the GS sends these updates affects the protocol's performance significantly. If the periodicUpdateInterval is low, the GS broadcasts updates more frequently resulting in increased control overhead. On the contrary, if the periodicUpdateInterval is high, the GS broadcasts updates less frequently resulting in the ANs not having up-to-date information about the other ANs. Velocities of all the ANs are unlikely to be uniform and some of the ANs may change their direction more frequently than others. Therefore, sending updates of all the ANs for every periodicUpdateInterval alone is not sufficient. There is a need for a mechanism where the ground station can broadcast updates in-between the periodicUpdateIntervals as well, called the trigger update mechanism. Whenever

the GS identifies a change in direction of any AN, it immediately broadcasts the changed location and velocity information of the AN. Highly dynamic changes result in the GS sending trigger updates more frequently that leads to more control overhead and increased packet collisions in the network. To avoid this, trigger updates over a time-interval specified by triggerUpdateInterval are aggregated into a single update and sent together. On the other hand, the GS verifies if a trigger update can be cancelled and the change be sent in the next periodic update. If the time duration to the next periodic update is less than the triggerUpdateInterval, then the trigger update is not sent.

To make sure the GS advertisements do not violate the maximum transmission unit (MTU) of 1500 B set by the MAC layer, they are fragmented to multiple packets. Each packet is uniquely identified by the time the topology table in the GS is updated and by a 16-bit fragment number.

4.2.2 Types of GS Updates

The ground station sends two types of advertisements: GSGeoLocation and GSTopology. The GSGeoLocation and GSTopology advertisements are explained in the following sub sections.

4.2.2.1 GSGeoLocation Advertisements

Geolocation information of all the nodes is advertised by ground station using GSGeoLocation advertisements. This advertisement carries multiple GSGeoLocationHeaders containing the geolocation coordinates and the velocity components in x, y, and z directions. Figure 4.2 shows the header format for GSGeoLocation-

Header. A ground station also maintains a topology table similar other ANs. The difference is that the ANs fill their table on receiving AeroRP updates, whereas the GS topology table is assumed to have updated information of the entire network based on the predetermined flight plans. The ground station uses this information to send out geolocation updates. It sets the EnableBroadcast flag to *true* so that this message can be rebroadcasted among all the ANs. Depending on the update mechanism used, the GS sends out updates for all the ANs or only for the ANs with changed information since the last update. The GS creates multiple geolocation headers, one for each node and adds them to a packet. It then creates a single TypeHeader, populates the necessary fields and adds it on top of the geolocation headers. The packet format for TypeHeader is shown in Figure 4.1. This packet is then broadcasted in the network. Every AN receiving the packet processes the type header, identifies the type of headers present in the packet and based on the header length it processes each header separately.

Geolocation updates have an option for adding start time and end time as well. These fields are used if an AN is flying in a pre-determined path. If the start time is populated, it is interpreted as the AN is located at the location coordinates specified by the geolocation header. The end time field is the time from when the ANs should stop using this location information. This field can also be set to next periodicUpdateInterval or the predicted time after which the AN might go out of the network.

4.2.2.2 GSTopology Advertisements

GSTopology advertisements carry multiple GSTopologyHeaders for all the links formed among nodes in the network. Figure 4.3 shows the header format for

GSTopologyHeader. Each GSTopologyHeader carries two 16-bit node-id addresses of the ANs forming the link, the start and the expire times for that link, and the link cost. The start and expire times are calculated based on node's geolocation and velocity information. A link is said to be established between two nodes if the euclidean distance between the two is less than their transmission range. The assumption here is that all nodes have the same transmission range. Based on the nodes geolocation coordinates the euclidean distance is calculated. The link expire time is also predicted based on the node's geolocation and velocity components. The expire time for an active link is increased until the euclidean distance between the new predicted locations of the two nodes is greater than their transmission ranges. GS calculates this information for all the possible links that can be established among all the nodes in the network. If there are n nodes in a network, considering the best case scenario where every node is connected to every other node, the total number of possible links are $n \times (n-1)/2$.

4.3 Processing of AeroRP updates

AeroRP uses a protocol id of 251 and works in conjunction with the AeroNP network protocol. The AeroNP protocol on receiving any packet from the MAC layer identifies an AeroRP packet by looking at the protocol id field in the AeroNPHeader, and if it is equal to 251, AeroNP delivers the packet to the AeroRP routing protocol along with the extracted AeroNPHeader. Each AN examines the sourceTimestamp field present in the AeroNPHeader. It compares this timestamp value with the stored, last-received timestamp value for that neighbouring node in its topology table. If the received timestamp is newer, the protocol updates its topology table with the geolocation information present in the AeroNPHeader. It

also updates the congestion and corruption indicators for the node from which this packet was received. AeroRP then compares its own GSTimestamp value with that of the neighbouring node's GSTimestamp value. If it identifies that the neighbour node does not have a newer update from the GS, it unicasts the GS updates that were received since the GSTimestamp of the neighbouring node. This mechanism ensures that every node has consistent information from the GS.

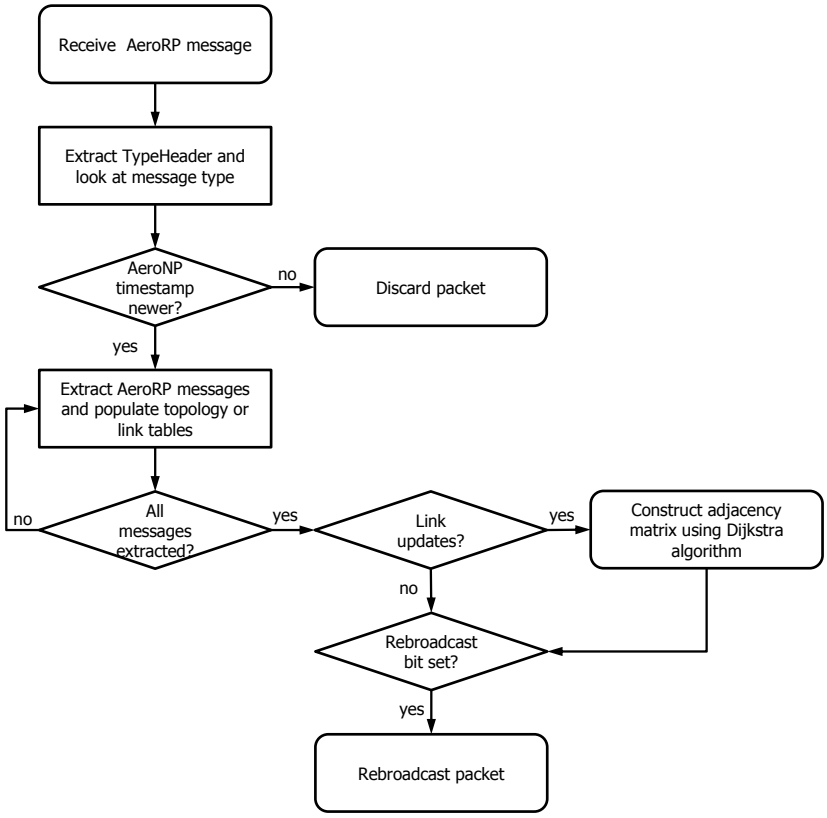

Figure 4.4. Flowchart for processing a GS advertisement

As explained in Sections 2.2.1 and 4.2, AeroRP uses three types of control messages to broadcast its updates. They are hello messages, GSGeoLocation advertisements, and GSTopology advertisements. The hello message is only used to inform the neighbours of a node about its presence. It uses the AeroNP header to

carry the node's geolocation information if AeroRP is operating in location-aware mode. AeroRP upon receiving a hello message, extracts the geolocation information from the AeroNP header and update its topology table. Let us assume a node A received a hello message from node B. If the topology table present in node A does not have any update for B, then A updates its topology table with the information received through the hello message and set the expire time as neighbourHoldTime. However, if A already has a GS update for B, it then verifies if B is still moving in the same direction and with the same speed as specified in the topology table. If true, node A does not update its topology table with the information present in hello message. If node A identifies that B has changed its direction or speed, it overwrites the information in the topology table with that received from the hello message and sets the expire time to neighbourHoldTime. In location-unaware mode, AeroRP verifies the link cost and if it receives a hello message with lower link cost, it updates its topology table and recomputes the routes using the Dijkstra shortest-path algorithm.

Upon receiving a GSGeoLocation advertisement, AeroRP unpacks all the GSGeoLocationHeaders one by one and updates the topology table entry for that node. The topology table is updated based in the timestamp at which the geolocation information is gathered. If the receiving node has a latest update from a node, it verifies if the GS update received for that node also predicted the node movement in the same direction and with the same speed. If true, the expire time for that node is updated with the expire time present in the GS update. GSTopology advertisements are also be processed the same way as a GSGeoLocation advertisements by extracting the GSTopologyHeaders. However, after all the GSTopologyHeaders are updated in the node's topology table, the protocol determines if the Dijkstra

shortest-path algorithm should be run based on the changed link state information. If there is a change in the link state information, AeroRP runs the Dijkstra algorithm, otherwise the algorithm is scheduled to run at the time determined by the creation of a new link based on the predicted geo-coordinates and velocity components. A timer schedules the Dijkstra algorithm by keeping track of the new link formations or breakages based on the information from the topology table. Figure 4.4 is the flowchart showing the GS update process.

4.4 Aero Gateway

Airborne communication data is expected to originate from a system supporting TCP/UDP/IP. This data should be moved over the domain-specific protocol suite and handed over to the destination which is again expected to be TCP/UDP/IP based. All the current iNET telemetry applications and devices are IP based which arises compatibility issues of the new protocol suite. To overcome this, we have introduced an interface called the aero gateway (AeroGW) [5], which resides on every node in the telemetry network including the GS. Data originating from, or destined to these applications is processed by the AeroGW to convert to the ANTP protocol suite. The AeroGW [5] translates the IP header to the AeroNP header and the TCP or UDP header to the AeroTP header. The original TCP/UDP/IP headers are removed from the packet and replaced with the newly generated AeroTP/AeroNP headers. Figure 4.5 shows the protocol stack architecture in the proposed ANTP protocol suite in the telemetry environment.

The AeroGW simulation in ns-3 was implemented with only the functionality of translating IP addresses used by the GS and the ANs to 16-bit device id used in the ANTP protocol suite. As the ANTP protocol suite is being simulated in

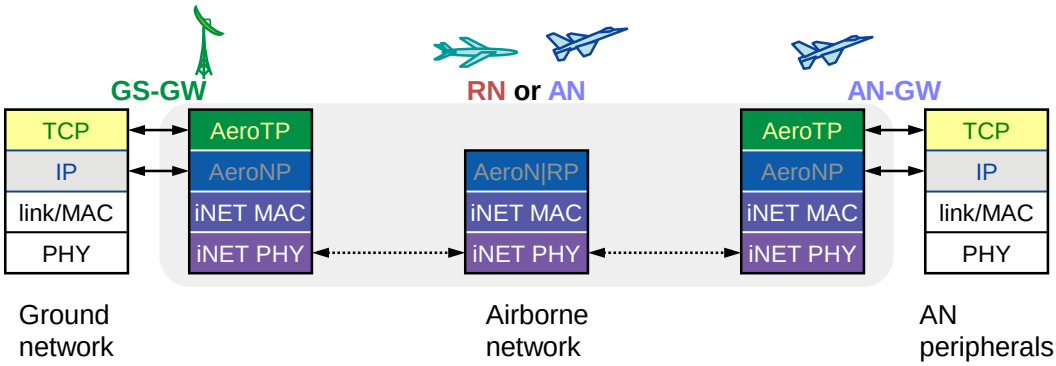

Figure 4.5. Protocol stack translation architecture ([5])

ns-3 that creates all sockets by binding them to IP addresses, it is necessary for the protocols being simulated in ns-3 to use IP addresses. To correctly evaluate the performance of the Aero protocols, the 32-bit IP addresses are mapped to a 16-bit device ids in these AeroGWs.

Chapter 5

Design of AeroNP

The AeroNP network protocol provides services to the AeroTP transport protocol as well as the AeroRP routing protocol. AeroNP encapsulates the packets coming from AeroTP and AeroRP protocols in the AeroNP protocol header. As mentioned in Section 2.2.2, AeroNP provides QoS, congestion-control and error detection services to the transport layer protocol. QoS is provided by maintaining a priority queues for the different levels of priorities specified by the mission plan. Congestion-control and error-detection mechanisms are implemented by maintaining a table that stores the congestion and corruption indicators of neighbouring nodes.

This chapter is organised as follows. Section 5.1 discusses the header format used by AeroNP. Sections 5.2.1, 5.2.2, and 5.2.3 briefly describe the QoS services, congestion-control, and error-detection mechanisms within AeroNP along with the design decisions made while implementing them in ns-3. The AeroNP protocol's packet transmission mechanism is explained in Section 5.3.2 and received packet processing mechanism is explained in Section 5.3.3.

5.1 AeroNP Header Format

The AeroNPHeader is of variable length due to the optional fields that can be added based on the location-aware or location-unaware routing employed. The optional fields that can be present in the AeroNP header are the transmitting node's geolocation information, destination node's geolocation information, and the latest GS timestamp and the fragment number present at the transmitting node. The geolocation information occupies 16 bytes of header space and the ground station's update information takes up 6 bytes. All these options are independent of each other. The presence of these options is indicated by the flags present in the header. AeroNPHeader with any of these options enabled is called AeroNPExtendedHeader and the one with none of these options enabled is called the AeroNPBasicHeader. The header length of AeroNPExtendedHeader with all options enabled is 60 bytes whereas the length of AeroNPBasicHeader is 20 bytes. Figures 5.1 and Figure 5.2 depict the AeroNPBasicHeader and AeroNPExtended-Header respectively. AeroNPHeaders are custom modified to suit the needs of ns-3 implementation. The field previous hop AN address in both headers and reserved field in the AeroNPExtendedHeader are added to resolve the ns-3 implementation issues. Each of the AeroNP header fields are explained in detail below.

- vers: 4 bits

  The vers field indicates the AeroNP's version number.

- CI: 2 bits

  The CI field indicates the congestion level for the type of traffic at the transmitting node's AeroNP queues. This field is used to regulate traffic flows in the network.

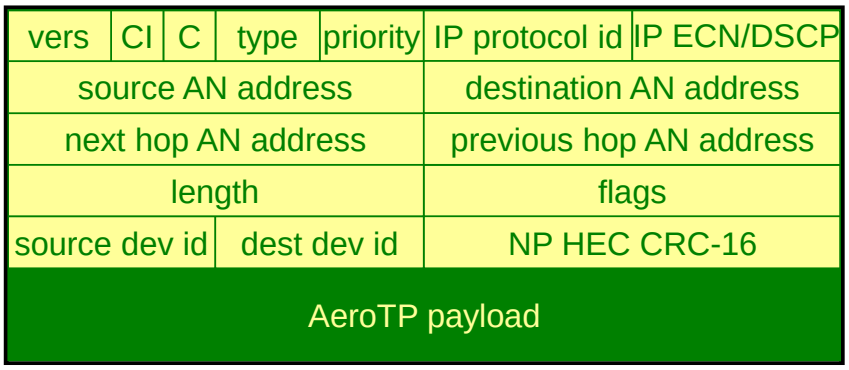

Figure 5.1. Packet format for AeroNPBasicHeader

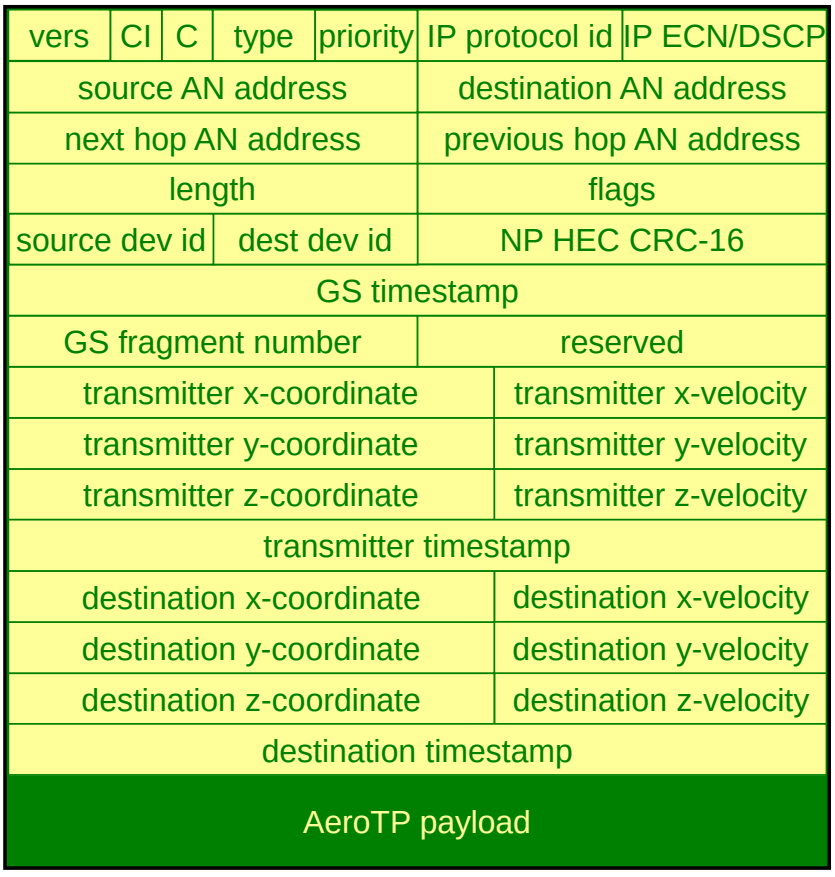

Figure 5.2. Packet format for AeroNPExtendedHeader

- C: 2 bits

  The C field indicates the packet corruption indicator at the transmitting node for the type of traffic data carried by the header.

- type: 4 bits

  This field specifies the type of traffic carried by this header.

- priority: 4 bits

  This field specifies the packet priority.

- protocol id: 8 bits

  This field specifies the upper layer's or the AeroRP's protocol id.

- IP ECN/DSCP: 8 bits

  This field carries the explicit congestion notification bits and the DSCP bits from IPHeader.

- sourceAN address: 16 bits

  This field specifies the source AN's 16-bit node id that transmitted this packet.

- destination AN address: 16 bits

  This field specifies the ultimate destination AN's 16-bit id.

- next hop AN address: 16 bits

  This field specifies the next hop AN's 16-bit id.

- previous hop AN address: 16 bits

  This field specifies the previous hop AN's 16-bit id. This field is specifically created to resolve implementation issues in ns-3.

- length: 16 bits

  This field specifies the AeroNPHeader length.

- flags: 16 bits

  This flags field is specifies the type of option fields carried by the AeroNP header.

- source dev id: 8 bits

  This field specifies the transmitter's interface id.

- dest dev id: 8 bits

  This field specifies the destination's interface id.

- NP HEC CRC-16: 16 bits

  This field is used to carry the CRC for error detection.

- GS timestamp [optional]: 32 bits

  This field specifies the ground station's timestamp that the node last received.

- GS fragment number [optional]: 16 bits

  This field specifies the ground station's fragment number that node last received.

- reserved [optional]: 16 bits

  This field added for word alignment after adding the previous hop AN address field. This field is only part of the AeroNP header used for ns-3 implementation.

- transmitter x-coordinate [optional]: 19 bits

This field specifies the transmitter node's x coordinate based on its current location.

- transmitter x-velocity [optional]: 13 bits

  This field specifies the transmitter's velocity component in x direction.

- transmitter y-coordinate [optional]: 19 bits

  This field specifies the transmitter node's y coordinate based on its current location.

- transmitter y-velocity [optional]: 13 bits

  This field specifies the transmitter's velocity component in y direction.

- transmitter z-coordinate [optional]: 19 bits

  This field specifies the transmitter node's z coordinate based on its current location.

- transmitter z-velocity [optional]: 13 bits

  This field specifies the transmitter's velocity component in z direction.

- transmitter timestamp [optional]: 32 bits

  This field specifies the timestamp at which the transmitter's geolocation information is recorded.

- destination x-coordinate [optional]: 19 bits

  This field specifies the destination node's x coordinate based on its current location.

- destination x-velocity [optional]: 13 bits

  This field specifies the destination's velocity component in x direction.

- destination y-coordinate [optional]: 19 bits

  This field specifies the destination node's x coordinate based on its current location.

- destination y-velocity [optional]: 13 bits

  This field specifies the destination's velocity component in y direction.

- destination z-coordinate [optional]: 19 bits

  This field specifies the destination node's x coordinate based on its current location.

- destination z-velocity [optional]: 13 bits

  This field specifies the destination's velocity component in z direction.

- destination timestamp [optional]: 32 bits

  This field specifies the timestamp at which the destination's geolocation information is recorded.

- AeroTP payload:

  AeroTP's payload is attached at the end of the AeroNP header.

5.2 Services provided by AeroNP

In this section we will briefly discuss the various services provided by AeroNP to the AeroTP transport protocol. We will also discuss about some of the design and implementation aspects of AeroNP.

5.2.1 QoS Mechanism

The packets coming from the AeroTP or AeroRP protocols are tagged with a priority value that can range from $0 - 3$. This priority value can be per flow or per application and that is determined by the mission plan. Drop-tail priority queues are maintained by AeroNP for each of the priority values. If a packet is not tagged with any priority value, AeroNP determines the priority of the packet based on the type of packet. The QoS services provided by AeroNP are thus based on type and priority of a packet. Upon receiving a packet, AeroNP enqueues the packet in one of the priority queues determined by the packet priority. Based on the priority scheduling algorithm employed, the packets are then retrieved from the queue and forwarded based on a route determined by the AeroRP routing protocol.

5.2.2 Congestion-control Mechanism

AeroNP provides congestion control service to AeroTP based on its neighbour's congestion indicator. AeroNP maintains a CCState table of the node's neighbours and their corresponding congestion indicator values for the various priority queues they maintain. A node advertises its congestion indicator by setting the CongestionIndicator bits in the AeroNP header. The neighbouring nodes operating in promiscuous mode capture the packet and update their CCState table with the congestion indicator value present in the AeroNP header. While determining a route for a packet with a specific priority, AeroRP selects the best next hop node by excluding the congested neighbours identified from this table for that priority. However if the congested node is the final destination for the packet, AeroNP forwards the packet to that node irrespective of its congestion level.

5.2.3 Error-Detection Mechanism

The AeroNP header has a field to store a 16-bit CRC used for detecting errors. Error detection *should* be implemented here as it helps in detecting packet errors at an early stage and rectify them either by resending the packet or trying to correct the errors.

5.3 Implementation of AeroNP in ns-3

AeroNP posed many issues during its implementation in ns-3. The AeroNP protocol does not identify nodes by their IP addresses but by the 16-bit node-ids. However the address format used by ns-3 is IPv4 and the sockets created in ns-3 are tightly bound to these addresses. Furthermore, protocol implementations are IP dependent in ns-3. So to plug-in AeroNP as a shim between IP and the transport layer posed another issue. In the current implementation of AeroNP in ns-3, AeroNP acts a layer-4 protocol that uses the services of IP. Figure 5.3 shows how AeroNP packets are encapsulated within IP in ns-3. Therefore, AeroNP should bypass IP's forward callback mechanism implemented in ns-3 and implement its own forward callback mechanism. To do this, the destination address in IP header is is always set to the gateway address which is the next hop address for the packet.

Figure 5.3. Header Encapsulation within IP

The IPHeader's destination address is always set as the next hop neighbour chosen by AeroRP. Thus AeroNP can choose what to do with the received packets, whether to deliver the packets locally to the transport layer if it is the destination

node or to forward packet to its neighbouring node. In the following section we will see how the various classes in AeroNP module interact and the packet transmission and receive mechanisms are implemented by AeroNP.

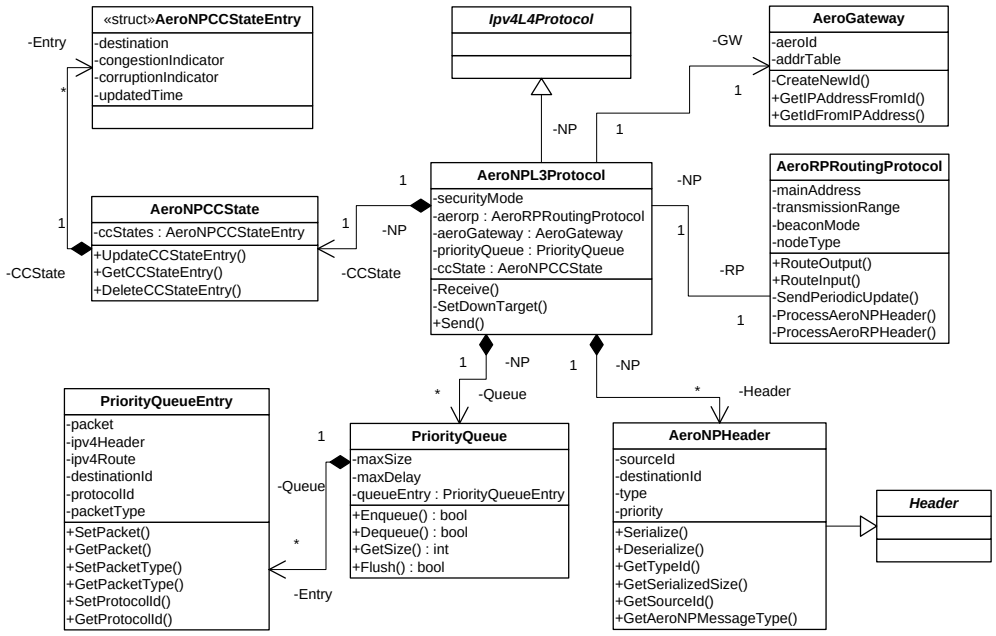

Figure 5.4. Class interaction diagram for AeroNP

5.3.1 Class interaction in AeroNP module

Figure 5.4 shows the various classes written as part of the AeroNP implementation in ns-3. `ns3::AeroNPL3Protocol` is the main class that takes care of all the AeroNP tasks with the help of other classes shown in this figure. It is derived from `ns3::Ipv4L4Protocol` base class. `ns3::PriorityQueue` implements the priority queues required by AeroNP to provide QoS. `ns3::AeroNPCCState` holds the congestion and corruption indicators of the node's neighbours to aid AeroNP in providing congestion-control and corruption-control services to AeroTP. `ns3`

::AeroNPL3Protocol takes services from the ns3::AeroRPRoutingProtocol and ns3::AeroGateway for identifying valid routes to destinations and for translating IP addresses to node ids and vice-versa respectively.

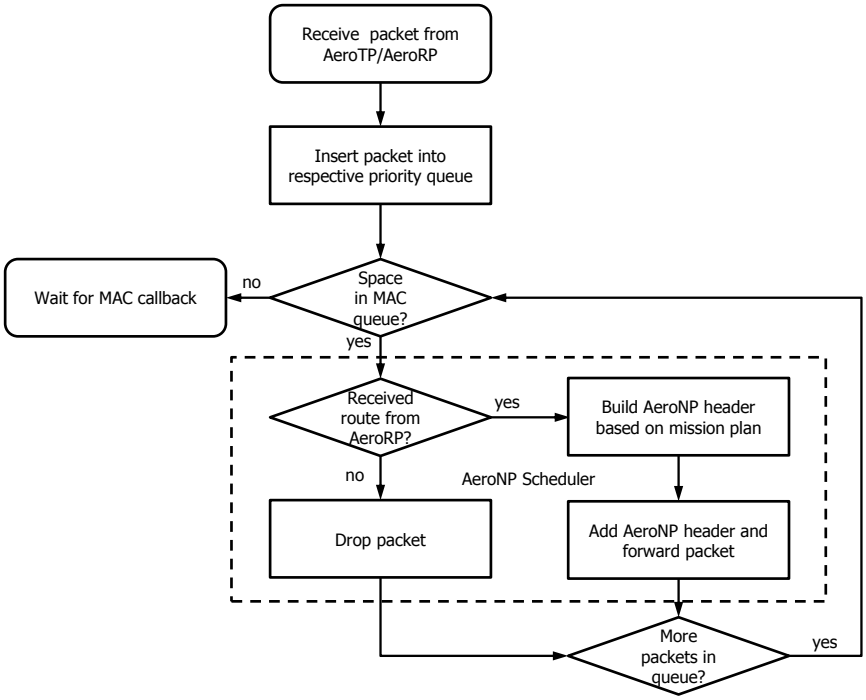

Figure 5.5. Packet transmission by AeroNP

When a simulation starts, ns3::AeroNPL3Protocol creates various instances of ns3::PriorityQueue depending on the number of priority levels specified by the mission plan and a single instance of ns3::AeroNPCCState. A maximum of 16 priority queues can be created as the AeroNP header uses 4 bits to carry the packet priority. ns3::PriorityQueue keeps a vector of pointers of type ns3::PriorityQueueEntry. It creates an instance of ns3::PriorityQueueEntry after receiving a request to queue a packet. Once the packet is dequeued, it de-

stroys the earlier created instance. `ns3::AeroNPCCState` also works in the same way except in this case, `ns3::AeroNPCCStateEntry` is a struct. `ns3::AeroNPHeader` that is extended from a generic `ns3::Header` class, holds all the code for creating both AeroNPBasicHeader and AeroNPExtendedHeader. An instance of this class is created whenever an AeroNP header is created.

5.3.2 Packet transmission by AeroNP

The process flow for transmitting a packet by AeroNP is depicted in Figure 5.5. `ns3::PriorityQueue` class stores all the packets coming from AeroTP or AeroRP. Along with storing the packets, it also stores source and destination addresses, IP-Header and AeroNPHeader if it is a forwarded packet, and the AeroNPMessageType. AeroNPMessageType helps in identifying the priority of packets. After buffering the packet, AeroNP invokes the AeroNPScheduler. This AeroNPScheduler can be invoked either directly by AeroNP or by the ns-3 callback mechanism from the MAC layer. This callback is invoked whenever the MAC dequeues a packet and transmits it. Based on the size of the `ns3::PriorityQueue`, the congestion indicator of the node is varied. AeroNPScheduler removes a packet from the queue based on the priority scheduling algorithm. The ns-3 simulation model of AeroNP currently employs a simple scheduling algorithm based on the priority value.

5.3.3 Received Packet Processing by AeroNP

The IP protocol transfers all packets received from the MAC layer to AeroNP. AeroNP on receiving these packets decides whether to forward a packet or deliver the packet locally either to AeroTP, AeroRP, or UDP. It makes this decision by looking at the destination address in the AeroNP header. If the packet is

destined for itself, AeroNP delivers it locally. If the received node is not the final destination, AeroNP moves the packet to the respective priority queue based on the type and priority fields in the packet. AeroNP scheduler is then invoked to identify a route to the destination as explained in Section 5.3.2.

Chapter 6

Simulations Analysis

The new GS update mechanism provides more options for ANs to select the best next hop neighbours and also provide a mechanism for the GS to broadcast strategic mission plans to the ANs. AeroRP with GS updates is implemented and simulated with ns-3 network simulator. ns-3 is a discrete event network simulator written in C++. The MANET routing protocols in the mainline release of ns-3 are DSDV [1], OLSR, and AODV. DSR routing protocol for ns-3 is being implemented by the ResiliNets group and will be part of the future mainline release of ns-3. In this chapter, we compare AeroRP and other MANET routing protocols in ns-3 by varying various parameters that affect their network performance. This chapter is organised as follows. The network performance metrics used for the analysis are detailed in Section 6.1. Section 6.2 briefly explains the different simulation parameters considered for this analysis. Section 6.3 analyses the variations in protocol's performance while running over a TDMA and an 802.11b MAC protocol. Section 6.4 analyses the protocols under varying node densities and Section 6.5 analyses them under varying node velocities.

[1] implemented as part of this thesis

6.1 Performance Metrics

The performance metrics considered for the evaluation of AeroRP are packet delivery ratio (PDR), accuracy, routing overhead, and delay.

- **Packet Delivery Ratio (PDR)**: The ratio of the number of packets received at the destination to the number of packets sent by the application. All packets sent down by the application are not be sent by the routing protocol if there is no route to the destination.

- **Accuracy**: Accuracy is the ratio of the number of packets received at the destination to the number of packets that were sent by the MAC layer. This is a good metric to gauge the quality of a route in a highly dynamic topology where the validity of a route can rapidly change.

- **Routing Overhead**: The fraction of bytes used by the protocol for AeroRP control messages. Overhead for data packets is calculated by subtracting the transport protocol's payload length from the IP header length. As for the control messages sent by routing protocols, the total IP header length is considered as the overhead. Thus overhead includes all the AeroRP control messages along with the AeroNP headers attached to every packet.

- **Delay**: The time taken by a packet to reach the destination node's MAC from the source node's MAC. Delay is calculated since the time it leaves the source node to the time it reaches the destination. It also includes the time the packet is buffered in the neighbouring node's queue.

The plots in the following sections detail the above metrics and contain confidence interval bars at the points in the plot. Since each simulation is run 10

times, the 95% confidence intervals are calculated using a t-distribution [40]. This is calculated as $M \pm A \times \frac{s}{\sqrt{n}}$ where M is the mean, A is the t-distribution value, s is the standard deviation, and n is the number of simulation runs for each point. The t-distribution value is 2.23 for 10 simulation runs with a 95% confidence. Note that some points may seem to not have any confidence interval bars. This is because they are too small to be seen on the plot. This indicates a higher confidence in the values that make up the mean.

6.2 Simulation Setup

This section highlights the various simulation parameters used for simulating these routing protocols in ns-3. Table 6.1 shows the parameters that are varied with all the routing protocols.

Table 6.1. Simulation variables

Variable	Values
Routing protocol	OLSR, AODV, DSDV, DSR, and AeroRP
AeroRP modes	GS–Location-aware, GS–Location-unaware, NotGS–Location-aware, and NotGS–Location-unaware
Node density	10, 20, 30, 40, 50, 60, 70, 80, 90, and 100 nodes
Mobility model	3D Gauss-Markov, Random waypoint, and Constant position
Velocity	10 m/s, 100 m/s, 200 m/s, 400 m/s, 600 m/s, 800 m/s, 1000 m/s, 1200 m/s, and a uniform distribution between 200 m/s and 1200 m/s
Link layer	TDMA and 802.11b

Table 6.2 highlights the general simulation parameters used for performing these simulations. All simulations are performed on ns-3.11 for a total simulation time of 1500 s. A warm-up time of 100 s is set so that the mobility models can reach a steady-state and the simulation is not affected by any initial conditions. However the warm-up time cannot affect the simulations using constant-position

mobility model as the nodes are immobile and are scattered randomly at the start of the simulation. Therefore the initial conditions in all simulations except for the ones using constant-position mobility model do not affect the outcome of these simulations as we are using the warm-up time. Constant bit-rate (CBR) traffic is sent from 100 s to 1100 s. A cool-down time of 400 s is set so that any packets that are buffered can be transmitted during this time. This ensures that all the CBR packets sent by a source has enough time to reach the destination. A transmit power of 50 dbm is chosen to achieve a transmission range of 27800 m (15 nautical mi).

Table 6.2. General simulation parameters

Parameter	Value
ns-3 version	ns-3.11
Number of times to run each simulation	10
Simulation area	150 km × 150 km × 1000 m
Initial position allocator	Random rectangle
Warmup time	100 s
Application sending time	1000 s
Cool-down time	400 s
Packet size	64 B
Sending rate	1 pkt/s
Packet fragmentation?	no
Propagation loss model	Friis
Transmission power	50 dBm
Transport protocol	UDP

All the OLSR routing protocol parameters are set the same as present in ns-3.11 except for the ones highlighted in Table 6.3. HelloInterval was changed from a default of 5 s to 1 s to suit the highly dynamic nature of this simulation environment. Similarly TcIInterval is set to 5 s and MidInterval is set to 5 s as well.

Table 6.4 highlights the parameters chosen for AODV routing protocol. Similar to the way OLSR routing protocol parameters were modified to suit the highly

Table 6.3. OLSR parameters

Parameter	Value
HelloInterval	1 s
TcInterval	5 s
MidInterval	5 s

Table 6.4. AODV parameters

Parameter	Value
HelloInterval	1 s
RreqRetries	2 retries for a route
RreqRateLimit	5 RREQ per second
NodeTraversalTime	40 ms
NextHopWait	50 ms
ActiveRouteTimeout	3 s
MyRouteTimeout	11.2 s
BlackListTimeout	5.6 s
DeletePeriod	8 s
NetDiameter	Number of nodes - 1
NetTraversalTime	2.8 s
PathDiscoveryTime	5.6 s
MaxQueueLen	500 packets
MaxQueueTime	30 s
AllowedHelloLoss	2 hellos
GratuitousReply	TRUE
DestinationOnly	FALSE
EnableHello	TRUE
EnableBroadcast	TRUE

dynamic nature of the simulation environment, some of AODV's parameters were also modified. RreqRateLimit was changed from its default value of 10 to 5. AODV model in ns-3 suffers from the RERR implosion problem. Though the ns-3 maintainers tried to rectify this issue, we still do not see any improvements in the performance of AODV. `https://www.nsnam.org/bugzilla/show_bug.cgi?id=1099` was opened to resolve this issue and it highlights the various issued faced with AODV. The maintainers insist that all the parameters are set as per the experimental AODV RFC [23].

Table 6.5. DSDV parameters

Parameter	Value
ForwardingInterval	4 s
SettlingTime	0 s
MaxQueueLen	Number of nodes × MaxQueuedPacketsPerDst
MaxQueuedPacketsPerDst	500 packets
MaxQueueTime	30 s
EnableBuffering	TRUE
EnableWST	FALSE
Holdtimes	3 × ForwardingInterval
EnableRouteAggregation	FALSE

DSDV routing protocols parameters are highlighted in Table 6.5. ForwardingInterval as modified from its default value of 15 s to 4 s and the SettlingTime was changed from 5 s to 0 s. With a SettlingTime of 0 s, DSDV uses a route in its perusal immediately without waiting to see if the route is stable or not. Also, buffering is enabled in DSDV with a maximum queue size set to 500 packets per destination.

Table 6.6 shows the DSR routing protocol's attributes and their values. NodeTraversalTime is the time a node waits for a passive acknowledgement or a reply from the neighbouring node indicating the successful transmission of the data packet that is set to 100 ms.

Table 6.6. DSR parameters

Parameter	Value
NodeTraversalTime	30 ms
PacketRetry	2
RouteCacheTimeout	30 s
MaxMaintTime	100 s
MaxMaintLen	500
MaxSendBuffLen	500
MaxSendBuffTime	100 s
PassiveAckTimeout	110 ms

Table 6.7. AeroRP parameters

Parameter	Value
Hello beacon interval	1 s
Neighbour hold time	4 s
Transmission range	27800 m
GSUpdateType	GeoLocationInformation and TopologyInformation depending on the type of routing mechanism used (location-aware or location-unaware)
GSUpdateInterval	20 s
GSTriggerUpdateInterval	5 s
GPSMode	TRUE (for GS) and FALSE (for ANs)
Ferry	TRUE
Transmission range	27800 m (15 nautical mi)

AeroRP sends out hello beacons for every 1 s if operating in beacon mode and if there is no data being sent out. It however does not send any beacons in beaconless mode. The GS update interval is set to 20 s and the trigger updates are aggregated over a GSTriggerUpdateInterval of 5 s. Table 6.7 shows the various parameters used for AeroRP routing protocol.

6.3 Analysis of TDMA vs 802.11

In this section we will analyse the effects of link layer protocols on the performance of the routing protocols. The link layer protocols used for these simulations are TDMA MAC protocol running on a simple-wireless channel and an 802.11b protocol running on YansWifiChannel model built in ns-3.

Figure 6.1 compares the variations in performance of DSDV, AeroRP with GS–Location-unaware mode, and AeroRP with NotGS–Location-aware mode running over TDMA and 802.11b. The variation in performance of the protocols is clearly visible in case of DSDV routing protocol. DSDV sends periodic updates for every

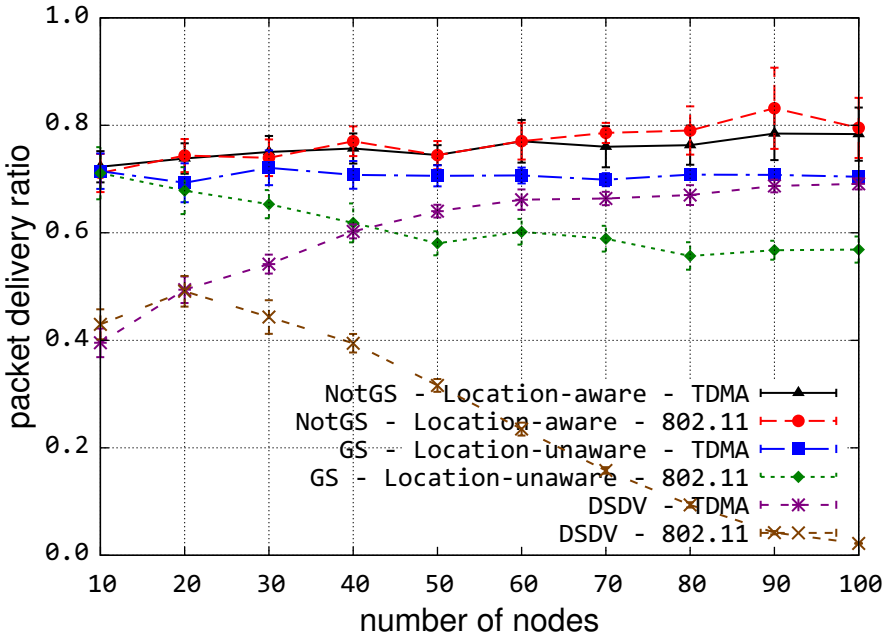

Figure 6.1. TDMA vs 802.11 on PDR (GM, 1200 m/s)

periodic update interval and trigger updates whenever it determines a change in its routing table. With the increase in number of nodes, the number of links in the network also increases thereby increasing the frequency of link state changes. Thus DSDV sends more and more updates as the number of nodes increases. Let us analyse the effect of this while using a TDMA MAC protocol. With the TDMA MAC protocol, every node has a specified slot allotted in a TDMA frame for it to transmit any packets. During this time all the other nodes will be in listen state. Hence there is no chance of collisions in TDMA network as opposed to the 802.11b network where every node simultaneously transmits packets leading to packet collisions in the channel. Though DSDV at every node takes care of sending these updates by waiting for a random time interval before transmission, most of the packet are lost due to collisions in the channel. In Figure 6.1, we can see that at 10 nodes, the performance of DSDV in TDMA and 802.11 networks is similar.

However with the increase in number of nodes, the packet delivery ratio of DSDV running on 802.11b drops but increases while using TDMA, which is expected as the network becomes more connected. This is the same with AeroRP running on GS–Location-unaware mode. In this mode, as the number of links increase with increase in the number of nodes, the GS sends more and more updates that get broadcasted among the ANs. Thus with the same reasoning applied for DSDV, AeroRP performs better on TDMA with the increase in number of nodes than when it runs over 802.11b. The PDR results for NotGS–Location-aware mode in both TDMA and 802.11 are similar as the overhead in this case is very less and both the MAC protocols are able to process all the packets generated by the AeroRP protocol.

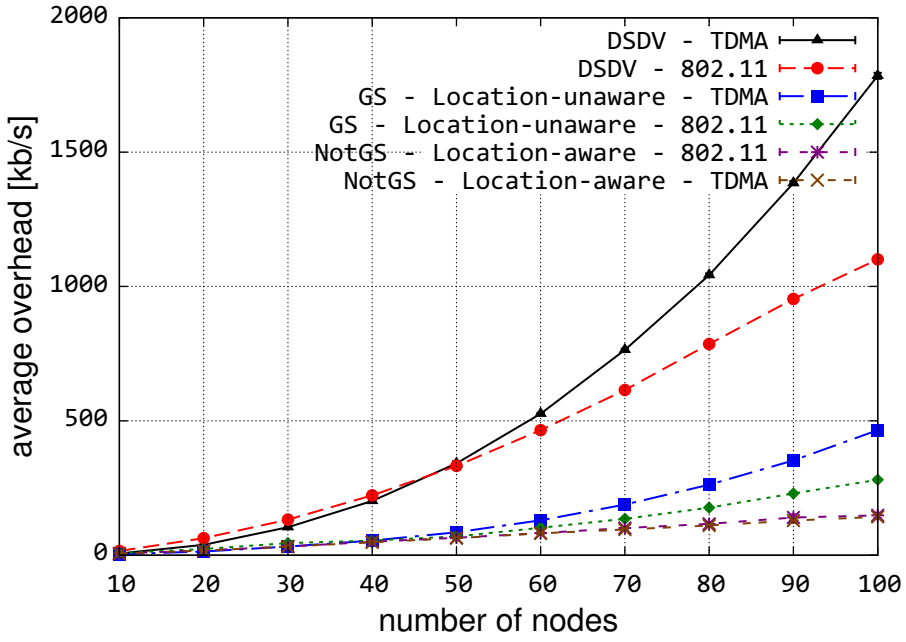

Figure 6.2. TDMA vs 802.11 on routing overhead (GM, 1200 m/s)

Figure 6.2 shows the variations in control overhead for DSDV and AeroRP with GS–Location-unaware and NotGS–Location-aware modes. We can see that the

control overhead is greater for all the protocols operating in TDMA as compared to them operating over 802.11. This is because more packets get through while using TDMA than while using 802.11. As most of the control packets get through, the nodes are aware of the changes in the network topology and thus they could recalculate routes to destinations based on these new changes. We can thus see an increase in the PDR in Figure 6.1 in which the protocols are running over TDMA. The overhead for NotGS–Location-aware mode in both TDMA and 802.11 is same as the overhead generated by AeroRP is significantly less and both the TDMA and 802.11 MAC protocols were able to transmit those packets without any packet loss.

Most of the analysis here is done by simulating these protocols on a TDMA MAC protocol as opposed to the 802.11b protocol. The main reason for using the TDMA MAC protocol for analysis is that the airborne networks run on a centralised TDMA MAC protocol developed by the iNet group. Furthermore, AeroRP being a geographic routing protocol, inherently keeps track of the node's transmission range and makes all its routing decisions based on this. However, with the current WifiNetDevice implementation in ns-3, a node's transmission range could only be limited by varying its transmit power. Thus a 50 dBm transmit power is chosen to have better throughput based on the research done in [6] for a 27800 m (15 nautical mi) transmission range. However this transmit power does not strictly limit the transmission range to 27800 m. Upon investigating, two nodes separated by a distance of 30000 m were also able to establish routes using OLSR. This is not a fair comparison between AeroRP that could not identify neighbours separated by more than 27800 m and OLSR that could identify neighbours even at 30000 m. The TDMA MAC protocol on the other hand is built over

a simple wireless channel model in which we can specify the desired transmission range as one of its parameters. This simple wireless channel model delivers all the packets to nodes that are within the specified transmission range. Also, the lower layer effects such as channel loss are not implemented in this channel model. We can thus analyse the performance of the upper layer protocols without worrying about the effects of the lower layers.

6.4 Effects of Node Density

In this section we will analyse the performance of protocols under varying node densities. As the grid boundary for the simulation area is fixed at 150 km $\times$ 150 km, the effects of the variations in node density can be analysed correctly with nodes confined to a particular region. The simulations are performed with node density varying from 10 to 100 nodes. The velocity of nodes is kept constant at 1200 m/s for the plots analysed in this section. Furthermore, the TDMA MAC protocol is used for the analysis as it removes the effects of the lower layers on the routing protocol performance as explained in Section 6.3.

Figure 6.3 shows the variation of PDR as node density increases from 10 nodes to 100 nodes. AeroRP in most of its modes has performed better when compared to DSDV, OLSR, AODV, and DSR. GS–Location-aware mode performs better compared to all other protocols with increase in node density. With the help of GS updates, AeroRP is able to have a full view of the network and is able to make better routing decisions. However the PDR for GS–Location-unaware mode drops slightly as the node density increases. At 1200 m/s, with the increase in node density, the number of links going up and down also increases. Thus the GS sends more and more trigger updates when it sees a change in the link state information.

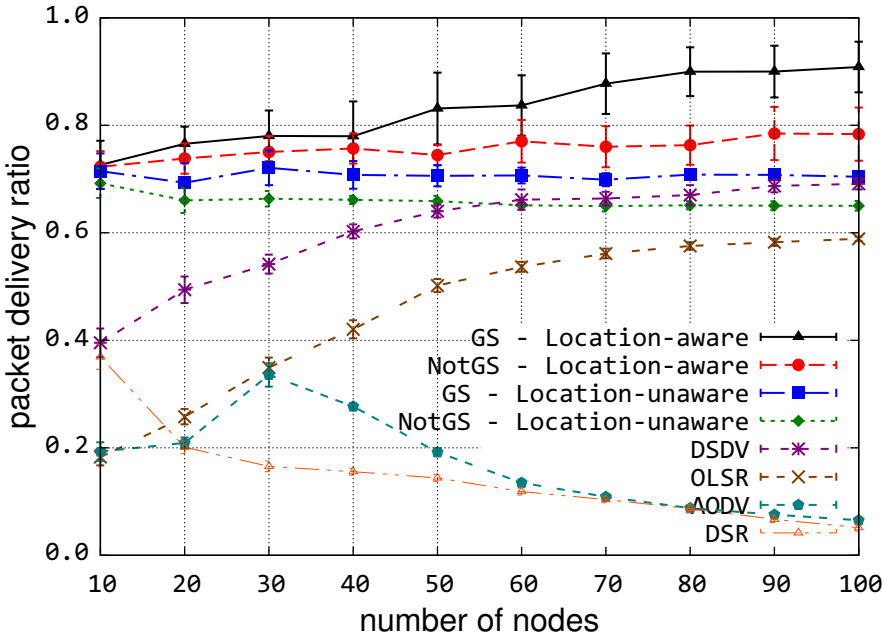

Figure 6.3. Node density vs PDR (TDMA, GM, 1200 m/s)

Further more, AeroNP gives highest preference to these AeroRP control packets over the data packets. Thus even though the network is more connected with the increase in node density, PDR for GS–Location-unaware mode remains nearly the same as there is significant control overhead sent by the GS. The PDR for DSDV and OLSR increases as the network is more and more connected with the increase in the node density. The main reason why GS–Location-unaware mode performs better at low node density when compared with either DSDV or OLSR is because of the use of store and haul mechanism [37].

Figure 6.4 shows the variation of control overhead with varying node density. We can see that the control overhead for DSDV increases more rapidly compared to other protocols as the node density increases from 10 to 100 nodes. At a high velocity of 1200 m/s, the link state information in the network changes more frequently with increase in node density. DSDV sends trigger updates whenever

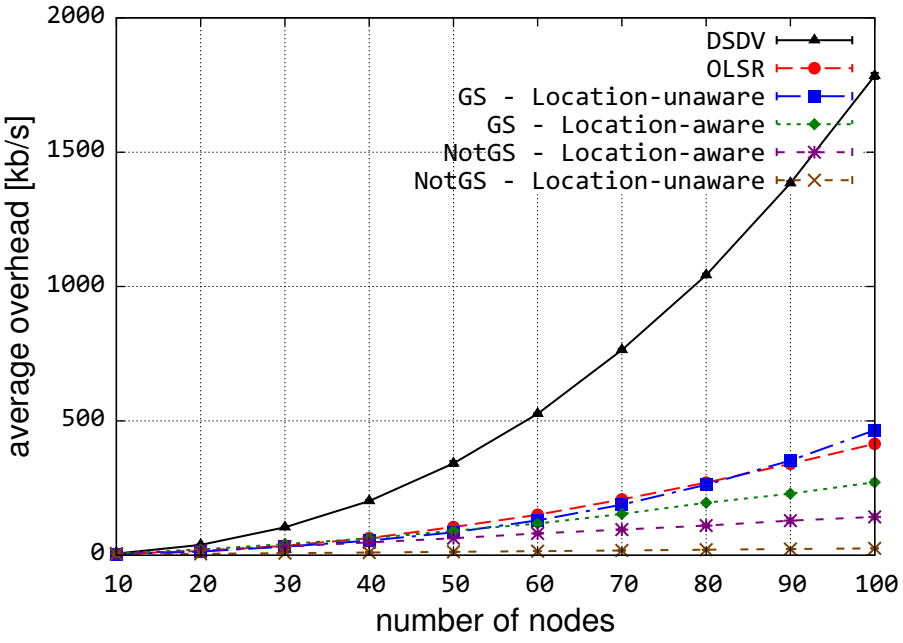

Figure 6.4. Node density vs overhead (TDMA, GM, 1200 m/s)

it identifies an existing link going down or when it identifies a new link with the shortest hop count metric. Thus the control overhead of DSDV shoots up with the increase in node density. The control overhead for GS–Location-unaware mode also increases as the node density increases which can also be attributed to an increase in the frequency of links going up and down among nodes. The GS tries to send trigger updates whenever it sees a change in the link state information in the network. But these updates are aggregated over an interval known as GSTriggerUpdateInterval reducing the number of updates propagated in the network. This mechanism ensures that the control overhead does not shoot up like it did for DSDV routing protocol. On the other hand NotGS–Location-unaware mode has a very low overhead. This is because in this mode, the nodes do not exchange any information among themselves and they do not add geolocation information to the AeroNPHeader. The location-aware modes with and without GS updates

show a slight increase in control overhead with the increase in node density. This is expected as with the increase in the number of nodes, the number of messages with geolocation information present in their AeroNP headers also increase. The control overhead for DSR and AODV is considerably more compared to other protocols as shown in Figure 6.5.

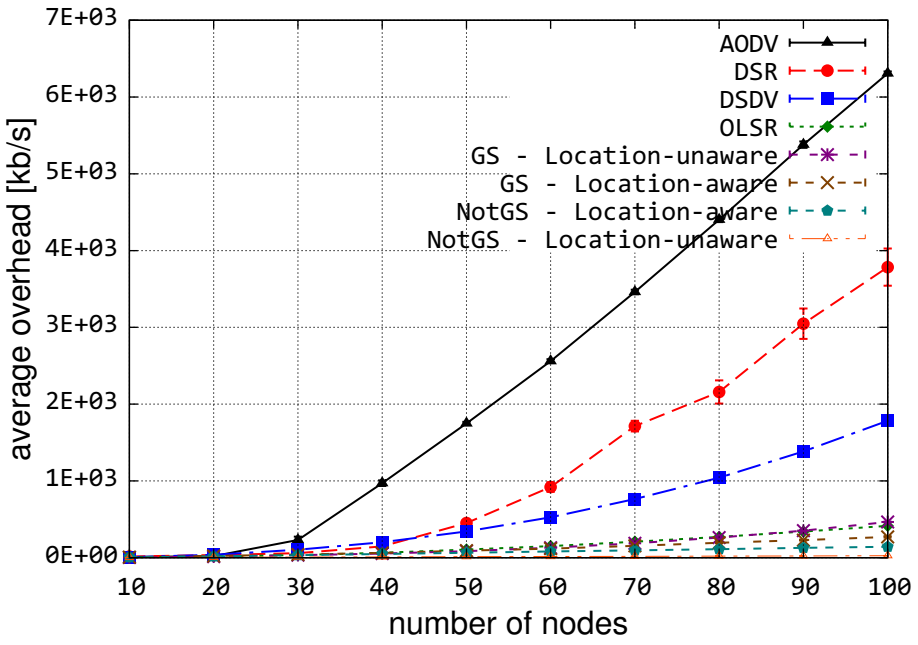

Figure 6.5. Node density vs overhead (AODV, TDMA, GM, 1200 m/s)

Figure 6.5 also depicts the variation in overhead as node density increases from 10 nodes to 100 nodes at 1200 m/s. The control overhead for AODV increases considerably with increase in node density and is thus plotted exponentially on the y-axis. AODV and DSR being reactive routing protocols, request a route whenever they receive a packet for transmission. Though the routes are cached, the link changes in the network are very frequent forcing the protocols to send more RREQ and RERR messages. The overhead for AODV is nearly 6 Mb/s at

100 nodes which is significantly higher than it should be. This issue was raised with the ns-3 developer team and we are working with them to identify if this is how AODV behaves or if there is some bug in the ns-3 AODV code. On the other hand, DSR has comparatively low overhead as it unicasts the RERR messages whereas AODV broadcasts them.

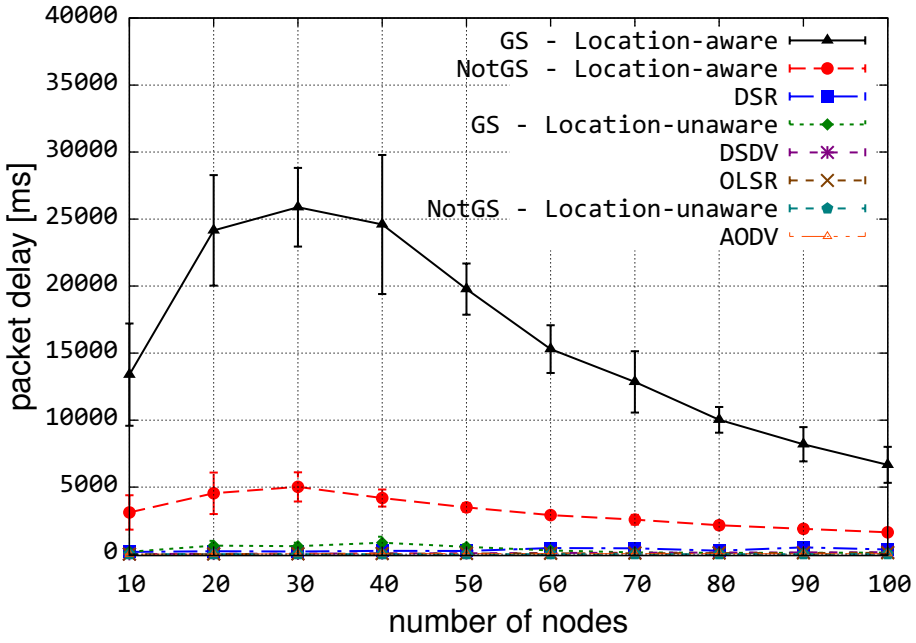

Figure 6.6. Node density vs delay (TDMA, GM, 1200 m/s)

Figure 6.6 analyses the change in packet delay with increase in node density. The packet delay in the location-aware modes of AeroRP is significantly higher compared to the AeroRP location-unaware modes and other MANET protocols. This is because of the time-to-intercept (TTI) metric used by the location-aware modes. The TTI metric predicts the amount of time a packet needs to be buffered before being able to find a route to deliver it to the destination. The location-aware mode can only choose a best next hop neighbour that is moving towards the destination, contrary to the location-unaware mode that can identify an end-

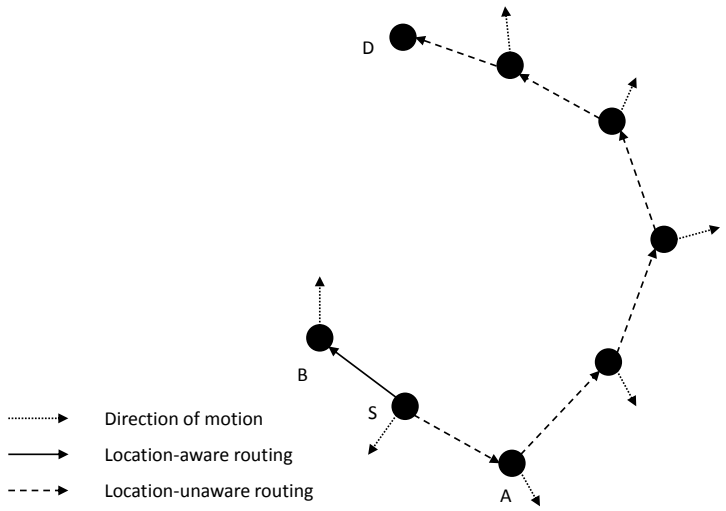

Figure 6.7. Location-aware vs Location-unaware mode in AeroRP

to-end route to the destination by selecting a next hop neighbour that is moving away from the destination. In Figure 6.7, source S wants to send a packet to destination D. S has two neighbours A and B with A moving towards the destination and B moving away from the destination. In location-aware routing, the TTI metric selects the best next hop neighbour as node A that is moving towards the destination. Node A then buffers the packet in its queue and delivers it to the destination upon reaching it. However in the case of location-unaware routing, AeroRP looks for end-to-end path and in this case selects node B that has a route to the destination. Thus the packet delay for location-aware routing modes in AeroRP is always more compared to the location-unaware modes. The packet delay for the other protocols such as DSDV, OLSR, AODV, and DSR is also very significantly as they also find the end-to-end path to a destination. We can see

that the packet delay for location-aware modes decreases as the number of nodes increase. This is because with the increase in number of nodes, the chances of identifying a node moving towards the destination also increases.

6.5 Effects of Velocity

The analysis of the variations in network performance caused by varying node velocities is important especially for highly dynamic airborne networks. In a typical airborne tactical network, nodes move at very high speeds and often the contact duration is less than 10 s. The increase in node velocity should not degrade the performance of the routing protocols. This is where the advantage of position based protocols is visible compared to other MANET routing protocols. AeroRP has many mechanisms built into it such as store and haul that are necessary for routing packets in this highly dynamic environment.

Figure 6.8 shows the variation of PDR as velocity is increased from 10 m/s to 1200 m/s. We can see that at low velocities the traditional MANET protocols perform better AeroRP. This is because at low velocities the links are quite stable and the end-to-end path is almost always stable. Location-aware modes of AeroRP do not perform well at low velocities. The 3D Gauss-Markov mobility model specifies a time step of 20 s after which every node changes its direction. Thus at a velocity of 10 m/s the node could only travel a distance of 200 m before it changes direction. There is a chance that with the new direction, the packet's initial source node is now moving towards the destination and the neighbour node that was moving towards the destination earlier is now moving away from it making the initial source node the best next hop neighbour. Furthermore, the 3D–Gauss-Markov mobility model of ns-3 [41] with the given set of parameters

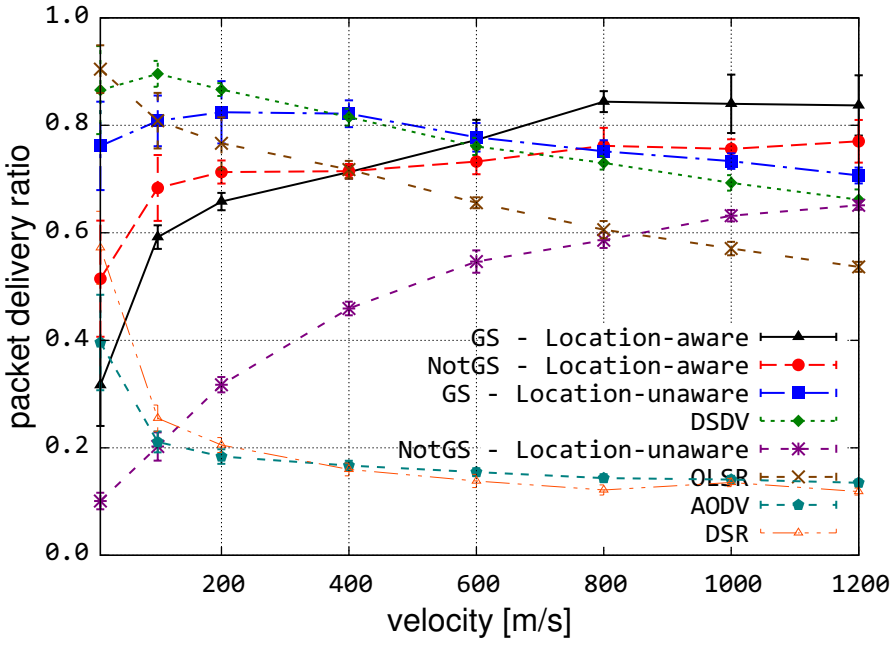

Figure 6.8. Node velocity vs PDR (TDMA, GM, 60 nodes)

makes the nodes move in nearly straight lines ($\alpha = 0.85$) and when they reach the simulation boundary, they usually take a near 180° turn and repeat the process. Thus the packets do not reach the final destination but come across dead-end situations. A dead-end situation occurs when a node does not have any other node to forward a packet or when it identifies the initial source node of the packet to be the best next hop neighbour. However as the velocity increases, the location-aware modes of AeroRP start to perform better than the other MANET protocols. With increase in velocity, the nodes move longer distances are are able to come in contact with many other nodes that may be moving towards the destination. As the node velocities increase, the traditional MANET routing protocols are not able to cope with increased frequency of links going up and down. This results in increased number of control messages transmitted over the network. The effects of increase in control overhead on PDR is lower since we are using a TDMA MAC

protocol on a simple wireless channel model. However, at high velocities the end-to-end path is not stable at any time and there is considerable packet loss that decreases the PDR for DSDV, OLSR, DSR, and AODV. GS–Location-unaware mode along with the other MANET protocols also tries to find the end-to-end path and thus faces the same issues faced by the MANET protocols.

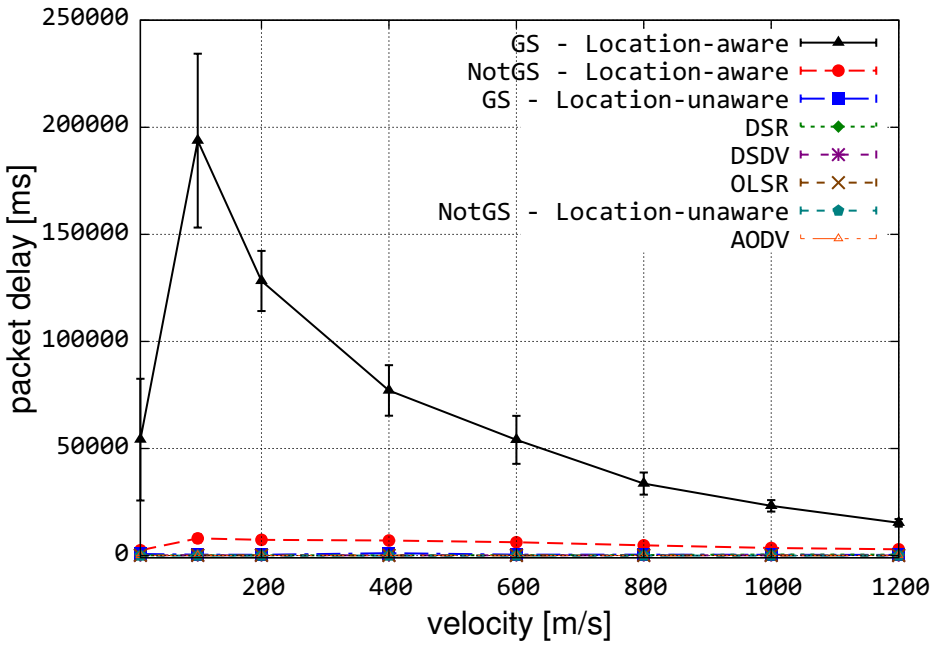

Figure 6.9. Node velocity vs delay (TDMA, GM, 60 nodes)

The packet delay plot shown in Figure 6.9 provides a very interesting analysis of AeroRP at low velocities and when it has information about the entire network. In location-aware routing, a node requires the neighbour's geolocation information to determine its TTI value. This geolocation information is exchanged only in the AeroNP header apart from the GS updates. A node could gather this geolocation information of its neighbours by snooping on the packets transmitted by them. However, a node operating in NotGS–Location-aware mode and moving towards the destination does not transmit any packets, rather buffers those packets in its

AeroRP queue as it has a better chance to deliver them. Any node not moving towards the destination tries to find the TTI of its neighbours but, it does not have geolocation information of any of its neighbours. So it buffers the packet in its AeroRP queue. In the GS–Location-aware mode, all nodes know the geolocation information of every other node in the network. So, they know their neighbours that are moving towards the destination in spite of those neighbours not transmitting any packets. Thus any node having a neighbour moving towards the destination forwards its packets to that neighbour expecting it to deliver those packets to the GS. A thing to note here is that, in the NotGS–Location-aware mode, the source node buffers the packets and in the GS–Location-aware mode the neighbouring node buffers the packets. The packet delay is the time interval from the time a packet leaves the source node to the time it reaches the destination. So packet delay in GS–Location-aware mode is larger compared to NotGS–Location-aware mode.

One more thing to observe here is that the packet delay is higher at low velocities (as high as 150000 ms) and as the velocity increases it drops to around 7000 ms. This can be attributed to the same reason why PDR is low at low velocities in location-aware modes of AeroRP. At low velocities, the chances of a node that is moving towards the destination and reaching it are minimal and the packets stay in a node's queue for a very long time increasing the packet delay. However, as the velocity increases, the chances of nodes communicating with other nodes is greater as they could travel longer distances before they change direction. So the delay decreases as there is more chance of the packet reaching the destination at high velocities. The packet delay for other protocols that look for full end-to-end path is low for obvious reasons as they do not transmit packet

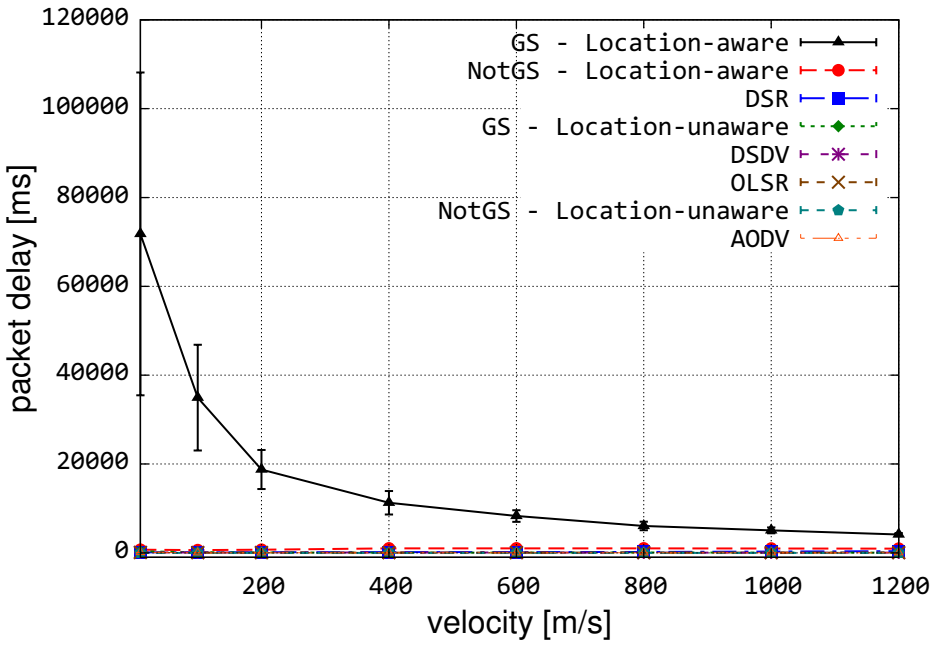

Figure 6.10. Node velocity vs delay (TDMA, RWP, 60 nodes)

unless they have a route to the destination.

Figure 6.10 shows the variation of packet delay using random waypoint mobility model [42]. We can see that the packet delay at lower velocities is much lower compared to the same using Gauss-Markov mobility model shown in Figure 6.9. The nodes in random waypoint mobility model choose a random position and move towards that position for a random time interval. During their movement, if they come across the simulation boundary, they choose a new direction and new time interval. Hence the nodes running random waypoint mobility model have more chance to come across other nodes during their movement and thus have better chance of delivering the packet to the destination in spite of moving at low velocities.

Accuracy analysis is also very important especially in highly dynamic environments as it determines how good a route identified by the routing protocol is.

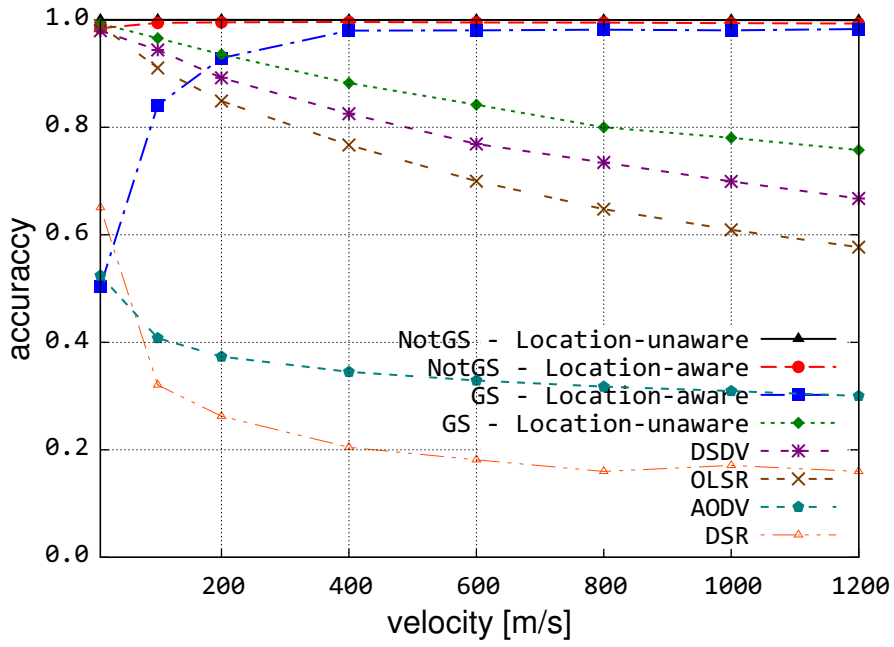

Figure 6.11. Node velocity vs accuracy (TDMA, GM, 60 nodes)

Figure 6.11 shows the variation of accuracy as the velocity increases from 10 m/s to 1200 m/s. The accuracy of the location-aware modes of AeroRP is 1.0 at almost all the velocities. The accuracy for GS–Location-aware mode at velocities less than 400 m/s is around 0.5. This is because of the way mobility models function in ns-3. At low velocities the there are more chances of reaching a dead end than there are to successfully deliver the packet to the destination. The nodes do not cover more distances at these low velocities as the mobility models change directions after a particular time interval. The movement pattern of Gauss-Markov mobility model is shown in Figure 6.12.

We can see that the nodes almost always move in straight lines and they come back. Let us consider a source node S forwarded a packet to its neighbour node A considering node A to move towards the destination. However with these low velocities, node A travels for some specific amount of time and they returns

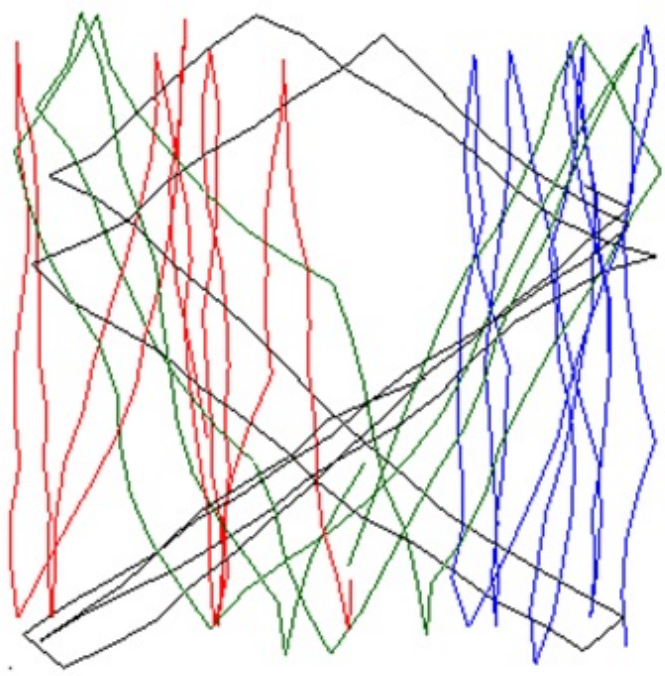

Figure 6.12. Node movement in 3D–Gauss-Markov model [6]

back. The chances of node A coming across the source node S in Gauss-Markov mobility model are greater as the nodes move in straight lines. When they meet, the neighbour node now is moving away form the destination and the source node S is moving towards the destination. Node A then forwards the packet to node S. However, as the packet initially was originated by node S, it drops the packet. This is most common dead-end case for AeroRP at low velocities. However in real airborne networks, the node movement will be somewhat different to what we see in these simulations. As the velocity increases, the probability of more nodes coming across each other increases thereby increasing the chances of transmitting packets to destination. Thus the accuracy of geolocation modes of AeroRP increases to 1.0 as velocity increases. Accuracy for protocols determining end-to-end path decreases as the velocity increases from 10 m/s to 1200 m/s.

As the velocity increases, the link-state information could change after an end-to-end path is determined and a packet is sent over it. The accuracy for the NotGS–Location-unaware mode is always 1.0 and it only has information of its neighbours at all times.

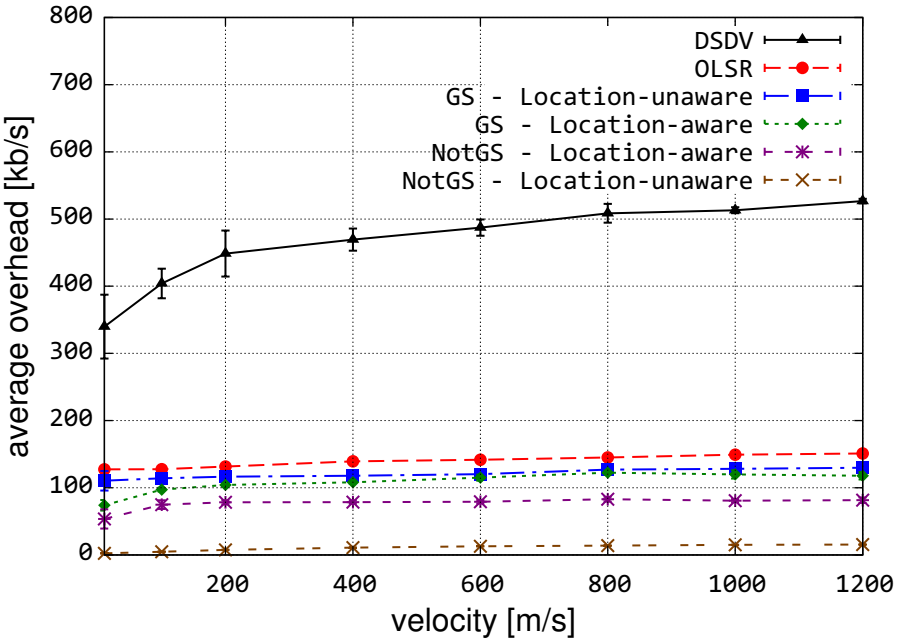

Figure 6.13. Node velocity vs overhead (TDMA, GM, 60 nodes)

Figure 6.13 shows how the control overhead varies as node velocity changes from 10 m/s to 1200 m/s. We can see that the control overhead remains nearly constant in the absence of GSAs as the velocities increases from 200 m/s to 1200 m/s. However, in the presence of GSAs the overhead slightly increases. This increase is more evident in GS–Location-unaware routing as the GS sends more trigger updates with increase in the frequency of change in link state information as node velocity increases. Furthermore, the number of trigger updates sent by the GS are decreased by the use of GSTriggerUpdateInterval that aggregates all the trigger updates into a single update. Every node in DSDV has to broadcast

an update whenever its link state information changes as opposed to only the GS broadcasting this update in AeroRP. Furthermore, every other node receiving that information, updates its routing table and broadcasts this information again. This may create a broadcast storm and thus the DSDV control overhead shoots up to nearly 500 kb/s at velocity of 1200 m/s. The overhead for AeroRP modes in the absence of GSAs is comparatively less than the other protocols and only slightly increases with increase in node velocity. In the absence of GSAs, the only overhead in the network is the AeroNP header information.

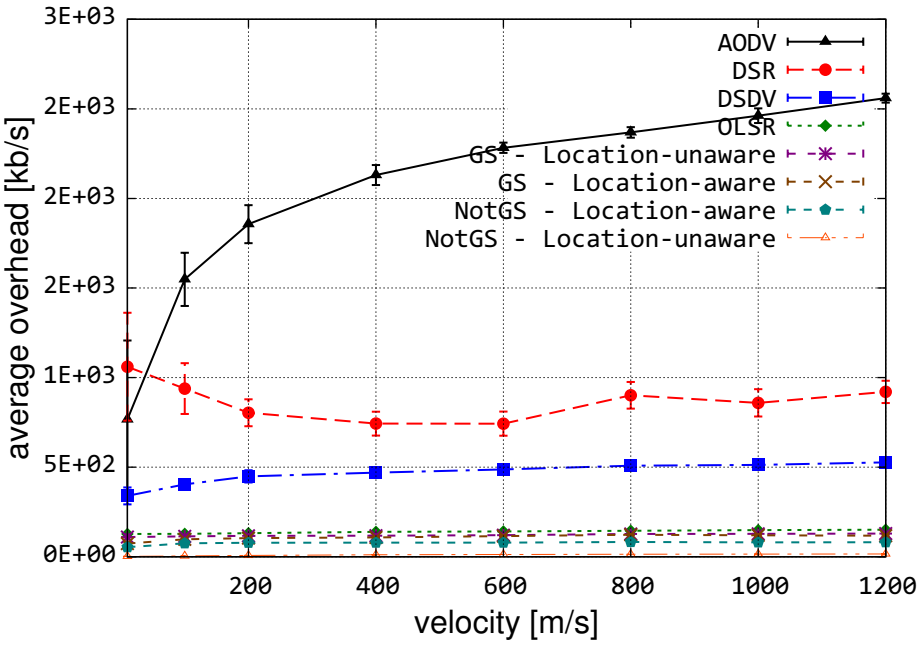

Figure 6.14. Node velocity vs overhead (AODV, TDMA, GM, 60 nodes)

Figure 6.14 is the same plot as Figure 6.13 but with the control overhead results for AODV and DSR included. The control overhead for AODV increases rapidly with increase in velocity and is thus plotted exponentially on the y-axis. We can see that the overhead for AODV is around 500 kb/s at 10 m/s and increases

rapidly to around 2.1 Mb/s at 1200 m/s. As the velocity increases, the links among nodes go up and down more frequently thus forcing AODV to send out more RREQ and RERR messages that increase the control overhead.

Chapter 7

Conclusions and Future Work

This chapter provides the concluding remarks in Section 7.2 and highlights the advantages of having GS updates to improve the overall network performance. Section 7.4 considers the future work required to improve on the current design and implementation of AeroRP and port it to miniature models and test it real time.

7.1 Contributions

The contributions of this thesis are:

- Design the AeroRP message headers and model the protocol for improving its performance
  - Choose the protocol parameters that can be used to modify the protocol operation
  - Incorporate `GS` updates and device a methodology of broadcasting these updates

  - Modify the protocol to use geolocation and topology information broadcasted by the GS in making routing decisions

- Implement AeroNP network protocol in ns-3
  - Modify AeroNP headers to suit the implementation decisions
  - Implement the QoS and congestion control services provided by the AeroNP protocol

- Implement the GS update mode and location-unaware routing in AeroRP routing protocol and the AeroNP network protocol in ns-3 network simulator

- Implement DSDV routing protocol in ns-3 to compare against AeroRP

- Implement TDMA MAC protocol in ns-3 over a simple-wireless channel model

- Analyse the performance of AeroRP in its various modes of operation and compare its performance against other MANET routing protocols such as OLSR, AODV, DSDV, and DSR in ns-3

7.2 Conclusions

This thesis provides a new neighbour discovery mechanism for AeroRP using ground station updates. It also provides a mechanism for AeroRP to operate in the absence of geolocation information. This thesis provides an overall view of MANET routing protocols and analyses their performance in high velocity scenarios and compares them against AeroRP in ns-3 network simulator. Chapter 2 discussed the various MANET routing mechanisms and protocols at our disposal

and highlights the drawbacks of conventional topology-based MANET routing protocols against the position-based protocols especially in the highly dynamic airborne network environment. Chapters 4 and 5 provide details about the design of AeroRP with GS updates and their implementation along with the implementation details of AeroNP network protocol. AeroRP's performance in its different modes is analysed in Chapter 6.

The nodes in highly dynamic airborne networks move at very high velocities. At these high velocities where the contact duration among nodes is as low as 10 s, the routing protocols should not concentrate on establishing the routes, but rather be ready to send out data packets as soon as they come in contact with their neighbour. AeroRP does this job well as we have seen in the analysis. It is an opportunistic geographic routing protocol that can predict the node movement and identify its neighbours via the three neighbour discovery mechanisms such as GS updates, active snooping, and hello beacons.

At low velocities the GS–Location-unaware mode works better compared to the location-aware modes of AeroRP. The location-unaware mode determines the end-to-end path whereas the packets are buffered in the location-aware mode. Though the buffered packets will reach the destination ultimately, the delay involved is greater as the nodes move slowly. The variations in link-state information is also less at low velocities thus decreasing the frequency of sending GS updates. So it is suggested to operate AeroRP in the GS–Location-unaware mode at low velocities. However as the velocities increase, the location-aware routing performs better. With the increase in velocity, there is more chance of the nodes coming in contact with each other thereby increasing the chances of delivering the packet to the destination. Furthermore, we have seen that as the velocity increases,

the accuracy of the protocols using the end-to-end path to perform routing has decreased.

The parameters for AeroRP must be chosen carefully depending on the network conditions and the mission requirements. The packet delay for AeroRP operating in the location-aware mode is much larger compared to the location-unaware mode. Proper care should be taken while selecting this mode for delay sensitive applications such as web traffic. On the other hand, the accuracy of packet delivery in location-aware mode if much higher than the location-unaware mode. The routing overhead can also be controlled by varying the GSPeriodicUpdateInterval. The frequency of periodic updates can be reduced if the nodes are moving at low velocities and can be increased if the nodes are moving at high velocities that results in often change in node direction.

OLSR sends Tcl messages for every 5 s and DSDV sends periodic updates for every 4 s. A change in link-state information will affect the routing tables in these protocols. Thus as the velocities increases, OLSR and DSDV have to send more and more updates thereby increasing the control overhead. PDR for OLSR and DSDV also drops and the end-to-end paths are not stable at high velocities. However, AeroRP operating in GS–Location-aware mode need not send periodic updates as frequently since the ANs can predict the path based on the last update received from the GS. As for the GS–Location-unaware mode, the updates can be aggregated over an interval called GSTriggerUpdateInterval and sent as a single update. This reduces the number of broadcast messages sent by the GS. Furthermore, AeroRP implements the store-and-haul mechanism; even if ANs do not have a route, they can buffer the packet until they can identify a route.

The reactive routing protocols AODV and DSR perform very poorly in this highly dynamic environment, which is expected. These protocols upon request for a route will initiate a route request, wait for the route reply from the destination, and then forward the packet to the destination. During this process if any link in the end-to-end path breaks, the whole process will be repeated again. However, AODV apart from being a reactive routing protocol, also has a problem with the RouteError (RERR) message explosion. The ns-3 model of AODV sends out enormous amount of RERRs that occupies almost the entire bandwidth of the channel. We are working with the AODV developers to resolve this issue [43]. DSR on the other hand is currently under development and is not in the mainline release of ns-3. We are working with the developer within the ResiliNets group to resolve its issues before being released to the community.

7.3 Publications

This section highlights my publications over the course of my Masters program.

- Hemanth Narra, Yufei Cheng, Egemen K. Çetinkaya, Justin P. Rohrer and James P.G. Sterbenz, "Destination-Sequenced Distance Vector (DSDV) Routing Protocol Implementation in ns-3", in the *4th International ICST Conference on Simulation Tools and Techniques, Wns3 2011* March 25, Barcelona, Spain.

- Abdul Jabbar, Hemanth Narra, and James P.G. Sterbenz, "An Approach to Quantifying Resilience in Mobile Ad hoc Networks", *The 8th IEEE International Workshop on the Design of Reliable Communication Networks (DRCN 2011).*

- Justin P. Rohrer, Egemen K. Çetinkaya, Hemanth Narra, Dan Broyles, Kevin Peters, and James P.G. Sterbenz, "AeroRP Performance in Highly-Dynamic Airborne Networks using 3D Gauss-Markov Mobility Model", *In Proceedings of the IEEE Military Communications Conference (MILCOM 2011)*.

- Hemanth Narra, Egemen K. Çetinkaya, and James P.G. Sterbenz, "Performance Analysis of AeroRP with Ground Station Updates in Highly-Dynamic Airborne Telemetry Networks", in *the International Telemetering Conference (ITC) 2011*, Las Vegas, NV.

7.4 Future Work

There is scope for more work to be done on AeroRP along with the other protocols in ANTP protocol suite. AeroRP operating in location-aware mode may come across dead end situations in low velocity scenarios. The reason identified is due to the way mobility models work in a simulation environment. Furthermore, a valid end-to-end path is not selected even though one exists as location-aware modes only take TTI metric into consideration. The TTI metric at low velocity scenarios will specify a packet buffer time in the order of tens of seconds, which is unacceptable for some types of traffic (web transactions, multimedia, etc.). A limit on the TTI metric could be specified so that the packets will not be buffered for long durations. However, selecting the optimal limit is a challenging task; it depends on the application requirements and network topology and should be further researched. One more consideration is the creation of a hybrid mode in which AeroRP chooses the operating modes based on the network topology and node velocities. At low velocities it could operate in location-unaware mode and

at high velocities it could operate in location-aware mode. Though the main use case for ANTP protocols is not in low velocity scenarios, this should be further researched when AeroRP is ported onto miniature vehicles. It would also be good to see how AeroRP performs when some relay nodes are added to the network. With the introduction of relay nodes, the performance of AeroRP should improve significantly as they will be able to deliver GS updates more efficiently and also act as sinks to the data transmitted from the ANs. This would also decrease the packet delay significantly especially in the location-aware modes. Priority scheduling in AeroNP should be further analysed and tested by transmitting packets with different priorities. Furthermore, more efficient priority scheduling algorithms could be designed and implemented to improve the performance of AeroNP. The simulations for this thesis were performed by combining AeroRP and AeroNP. All the protocols in the ANTP suite such as AeroTP, AeroNP, AeroRP, and AeroGW should be simulated together and see how the performance of AeroTP is affected in the different modes of AeroRP.

Appendices

Appendix A

802.11b Plots

The following plots have similar results to the plots analysed in Chapter 6 but these plots are obtained from simulations running on 801.11b MAC protocol.

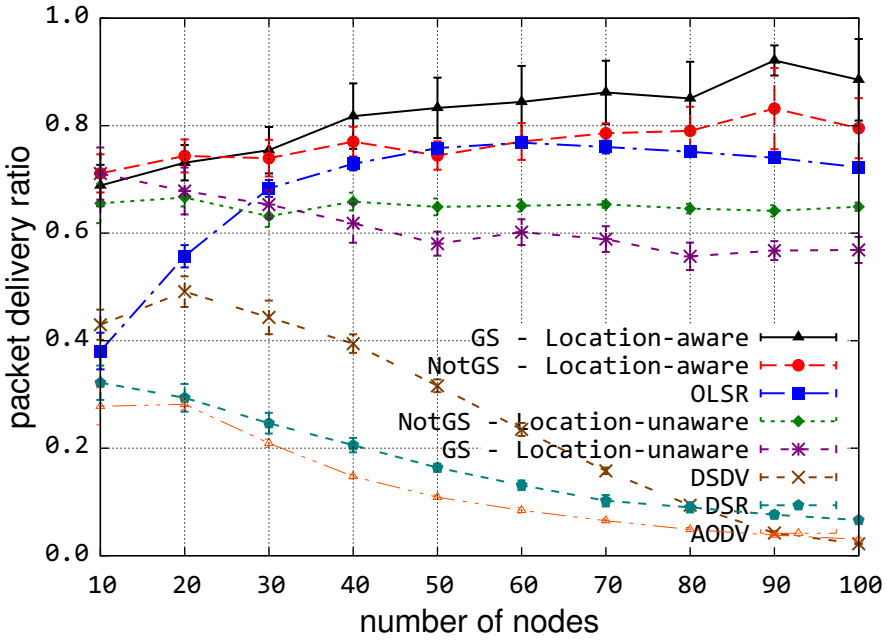

Figure A.1. Node density vs PDR (802.11b, GM, 1200 m/s)

In Figure A.1 we can see that OLSR performs slightly better over 802.11b MAC compared to the TDMA MAC as we have seen in Figure 6.3. This is because of the advantage OLSR has with not being able to strictly control the transmission range by specifying the transmit power in ns-3. OLSR was able to make more associations in the network with its higher transmission range whereas AeroRP was confined to 27800 m transmission range. DSDV's PDR decreased with increase in node density as its control overhead increased leading to generation of more packets that led to collisions in the network.

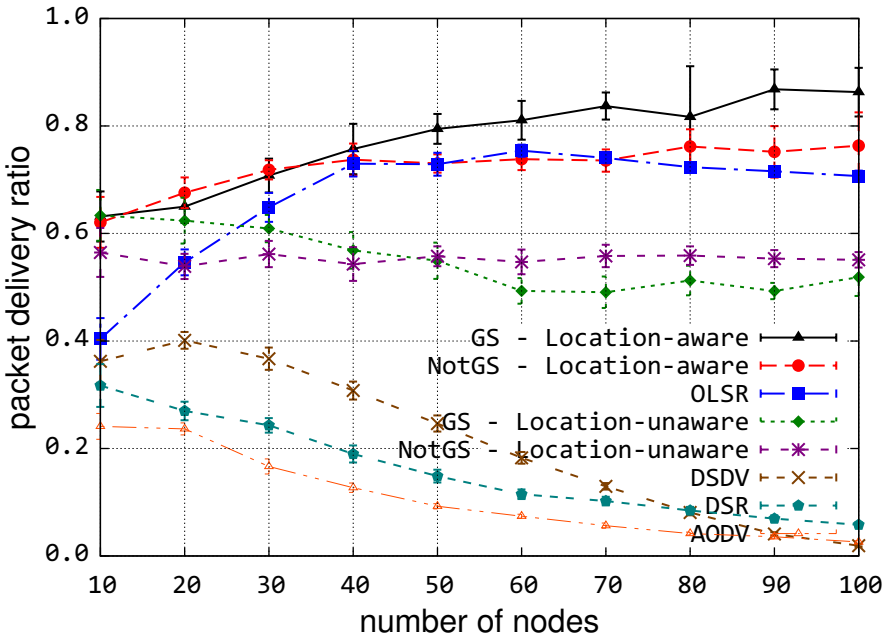

Figure A.2. Node density vs PDR (802.11b, GM, 200–1200 m/s)

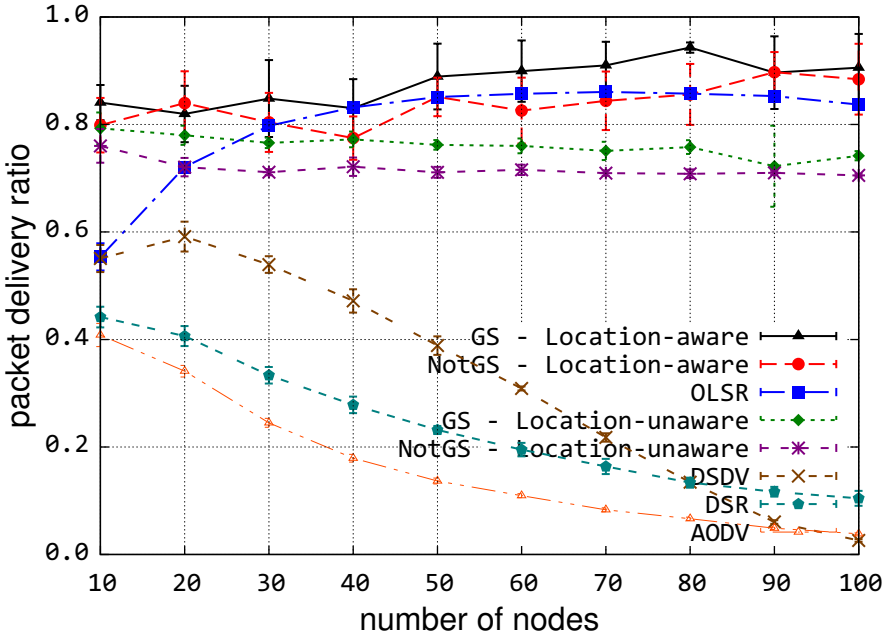

Figure A.3. Node density vs PDR (802.11b, RWP, 1200 m/s)

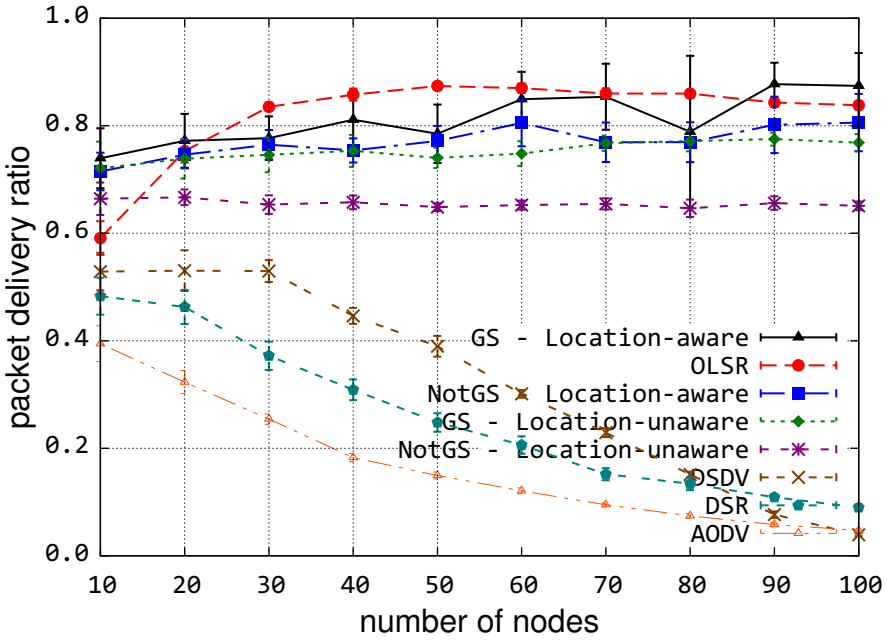

Figure A.4. Node density vs PDR (802.11b, RWP, 200–1200 m/s)

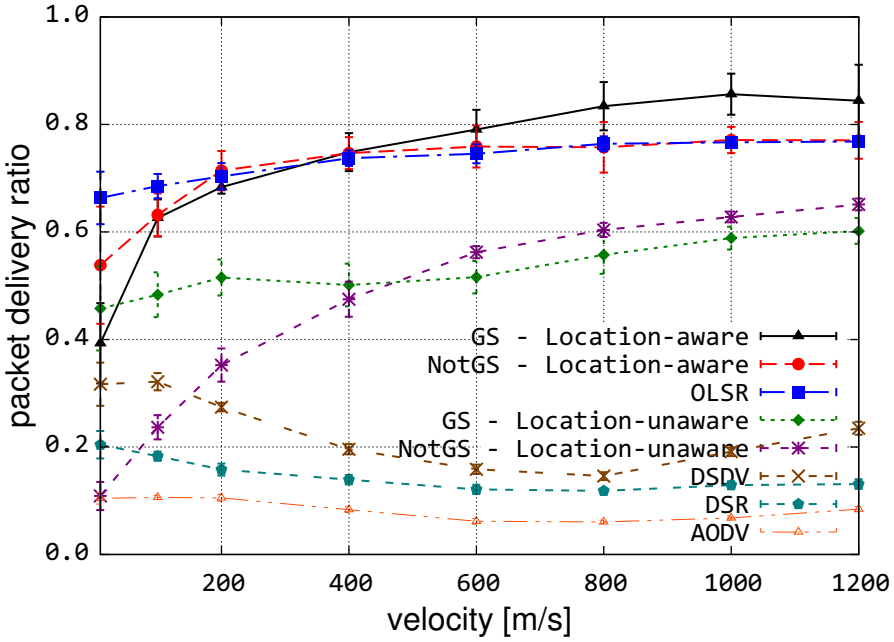

Figure A.5. Node velocity vs PDR (802.11b, GM, 60 nodes)

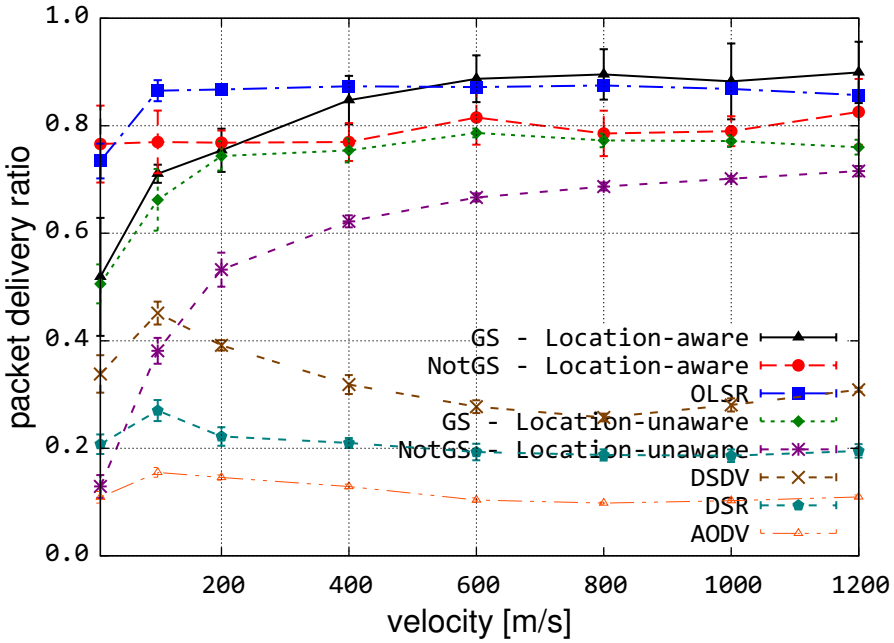

Figure A.6. Node velocity vs PDR (802.11b, RWP, 60 nodes)

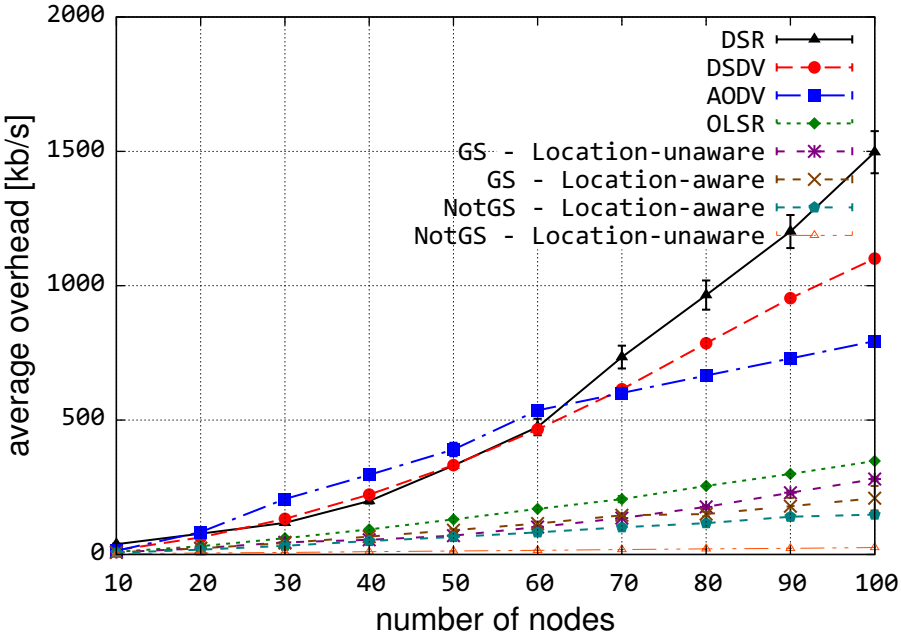

Figure A.7. Node density vs overhead (802.11b, GM, 1200 m/s)

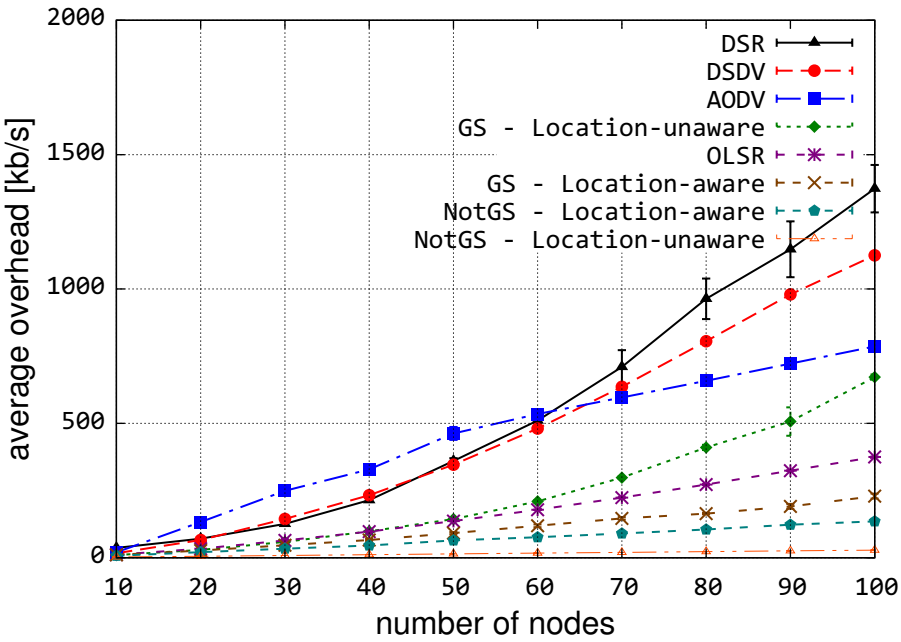

Figure A.8. Node density vs overhead (802.11b, RWP, 1200 m/s)

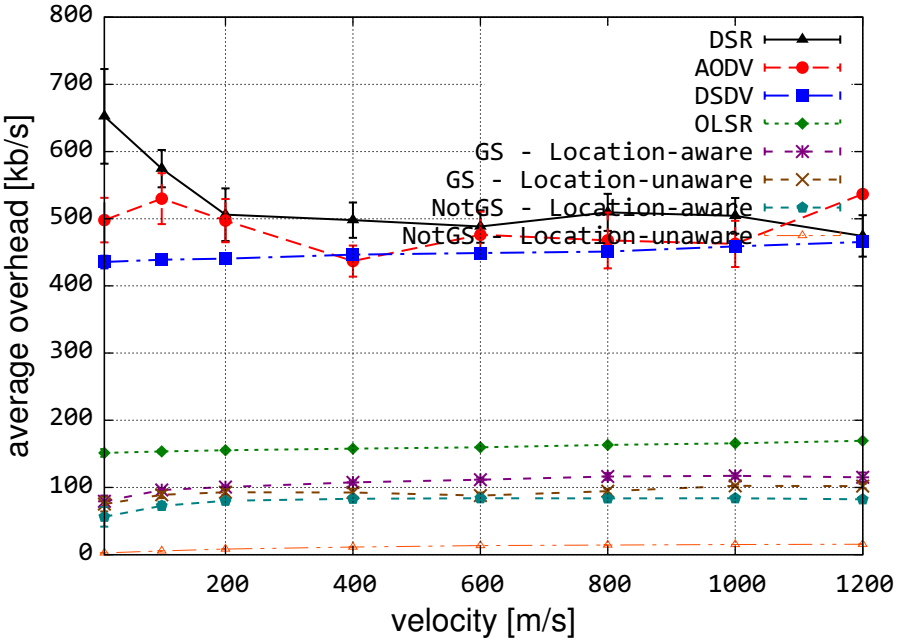

Figure A.9. Node velocity vs overhead (802.11b, GM, 60 nodes)

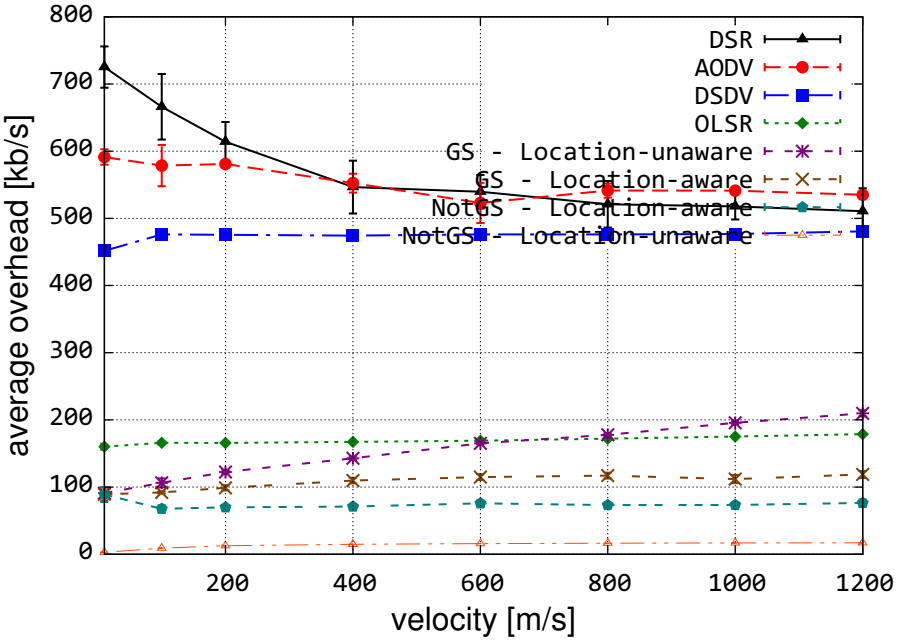

Figure A.10. Node velocity vs overhead (802.11b, RWP, 60 nodes)

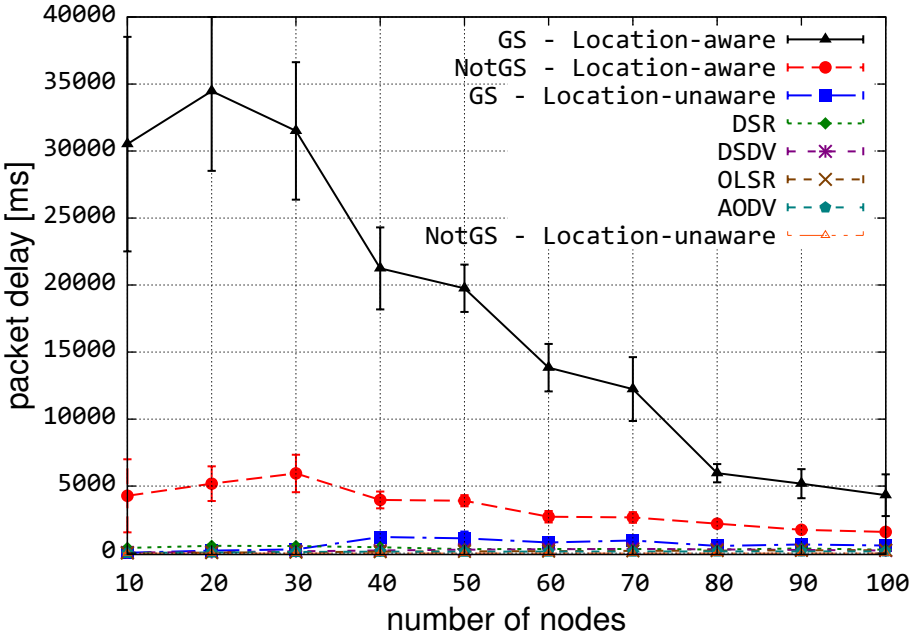

Figure A.11. Node density vs delay (802.11b, GM, 1200 m/s)

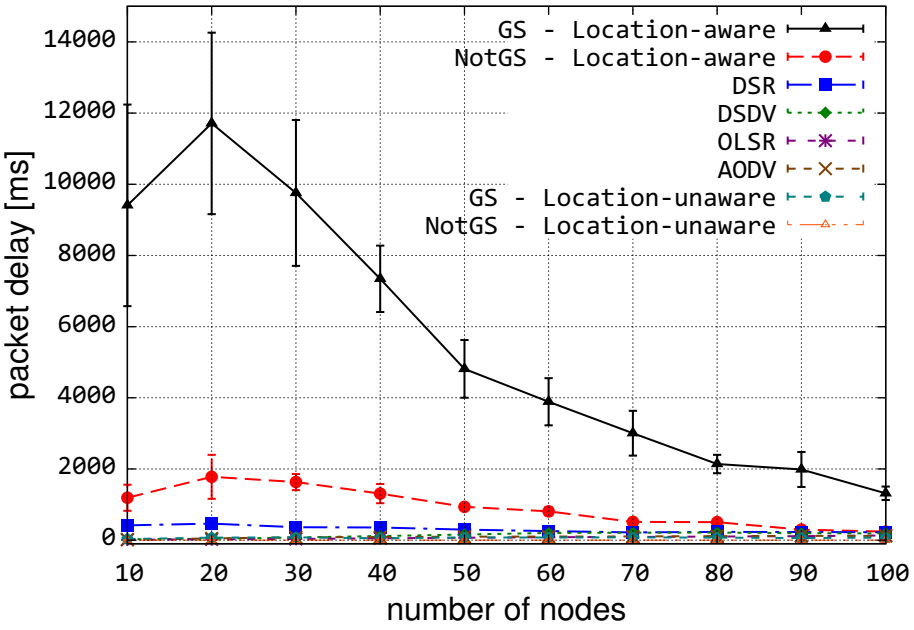

Figure A.12. Node density vs delay (802.11b, RWP, 1200 m/s)

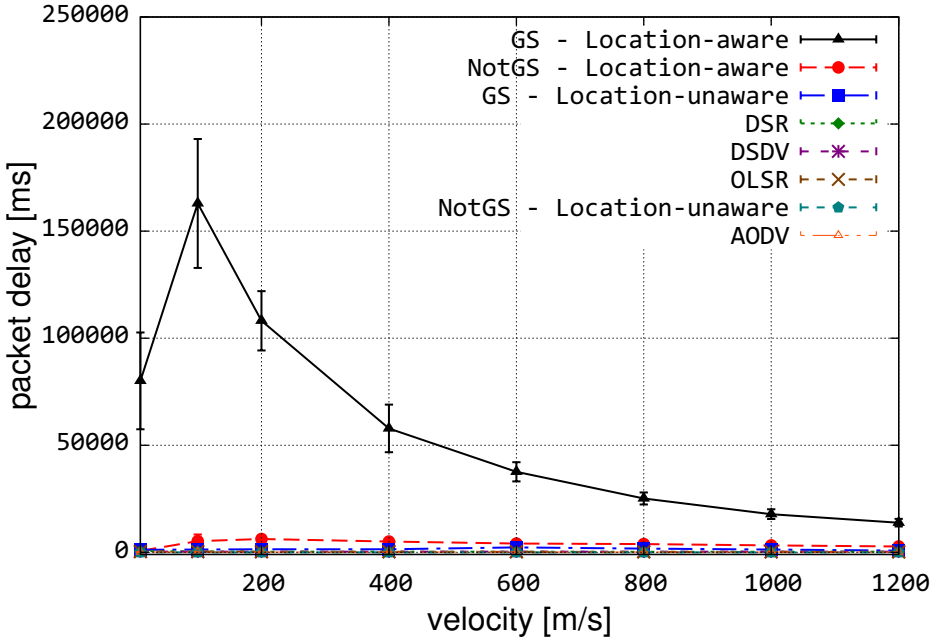

Figure A.13. Node velocity vs delay (802.11b, GM, 60 nodes)

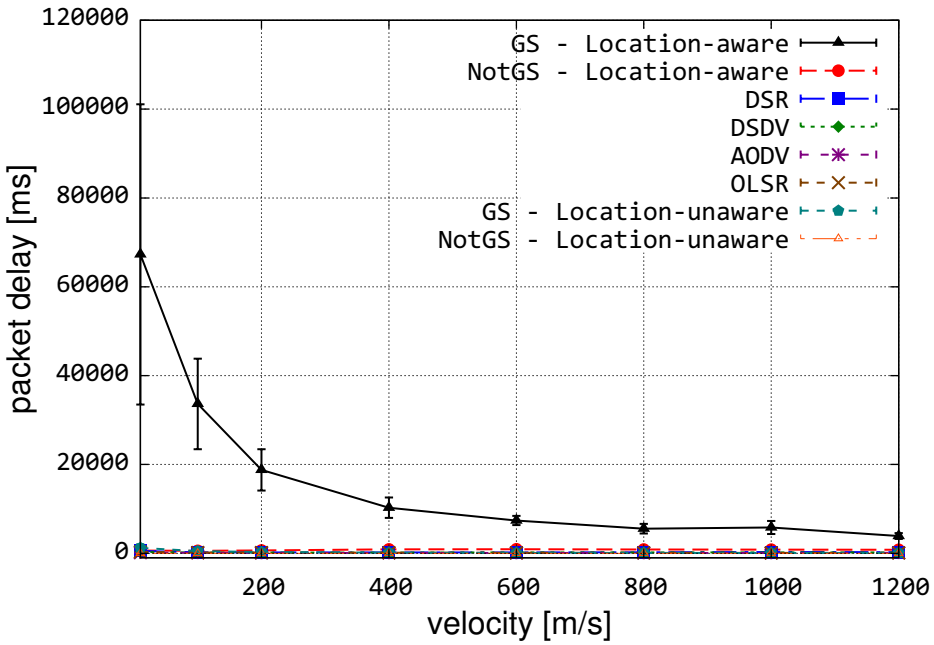

Figure A.14. Node velocity vs delay (802.11b, RWP, 60 nodes)

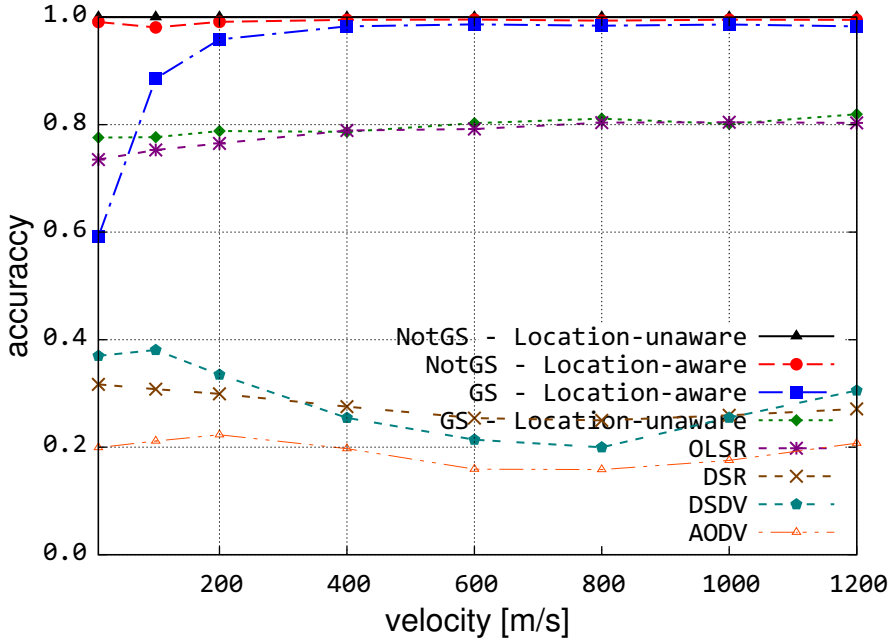

Figure A.15. Node velocity vs accuracy (802.11b, GM, 60 nodes)

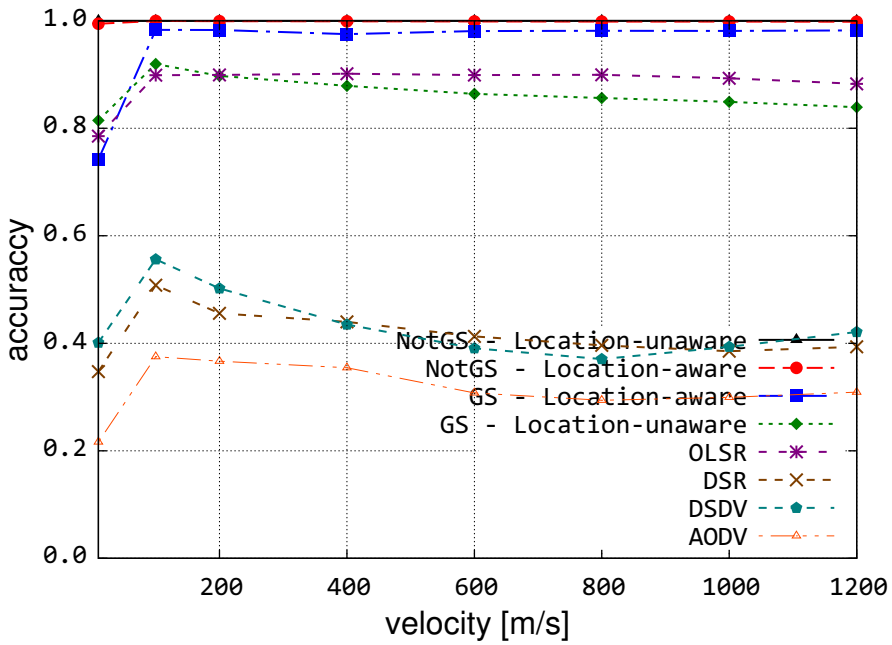

Figure A.16. Node velocity vs accuracy (802.11b, RWP, 60 nodes)

Appendix B

TDMA Plots

The following plots are obtained by performing simulations over Random way-point mobility model.

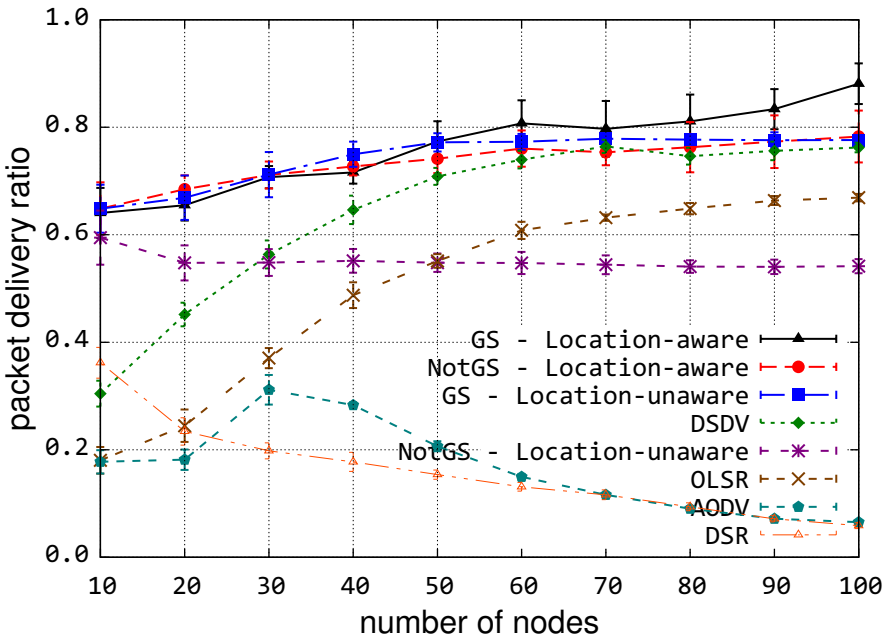

Figure B.1. Node density vs PDR (TDMA, GM, 200–1200 m/s)

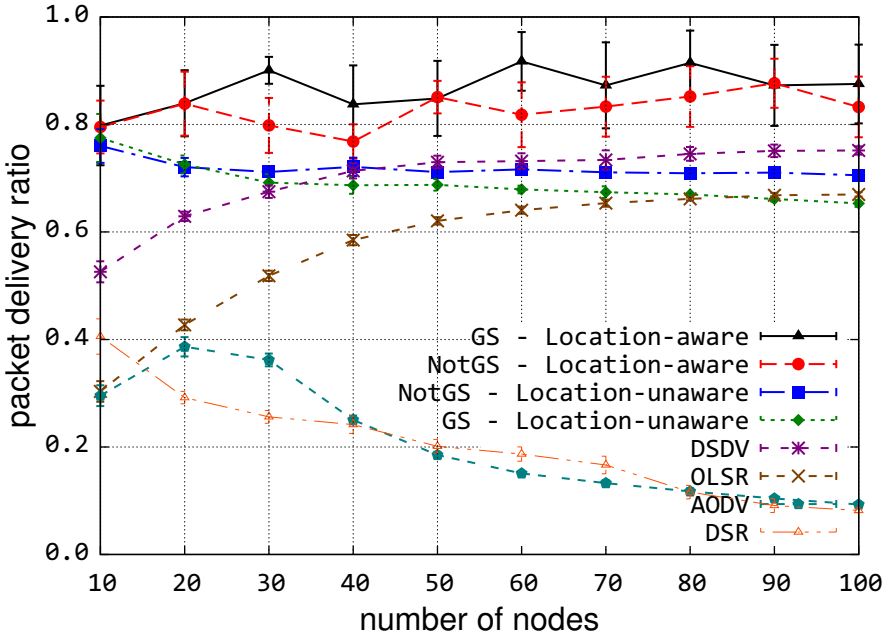

Figure B.2. Node density vs delay (TDMA, RWP, 1200 m/s)

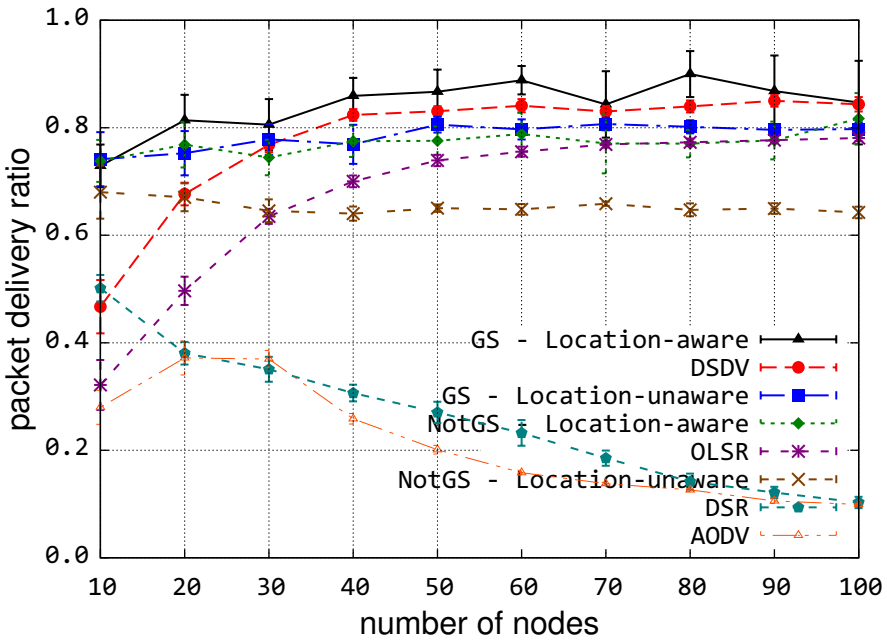

Figure B.3. Node density vs PDR (TDMA, RWP, 200–1200 m/s)

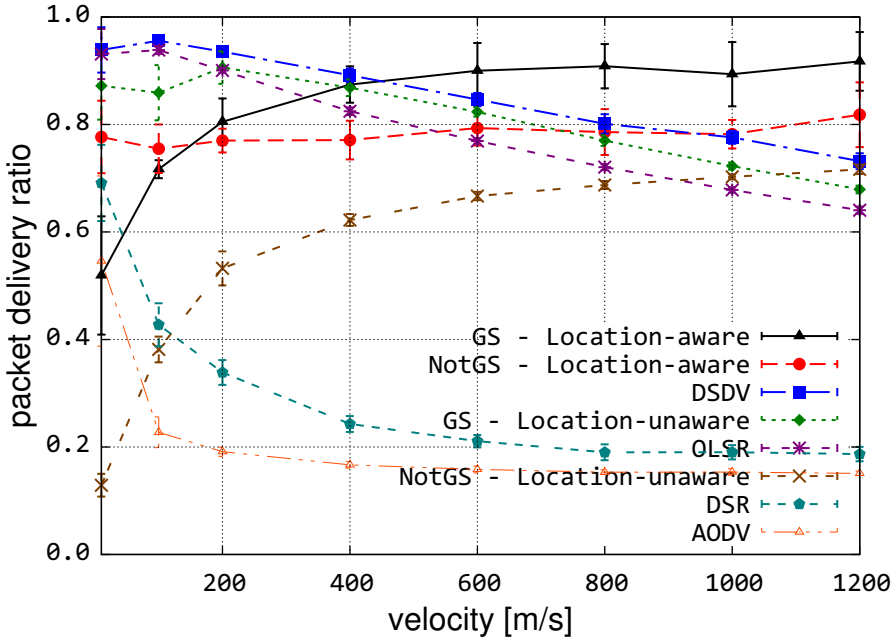

Figure B.4. Node velocity vs PDR (TDMA, RWP, 60 nodes)

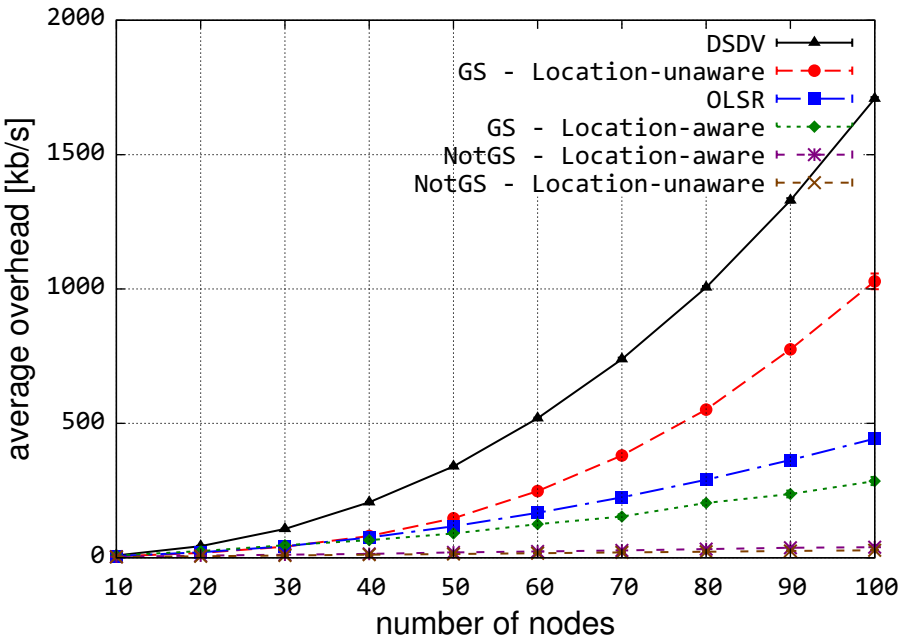

Figure B.5. Node density vs overhead (TDMA, RWP, 1200 m/s)

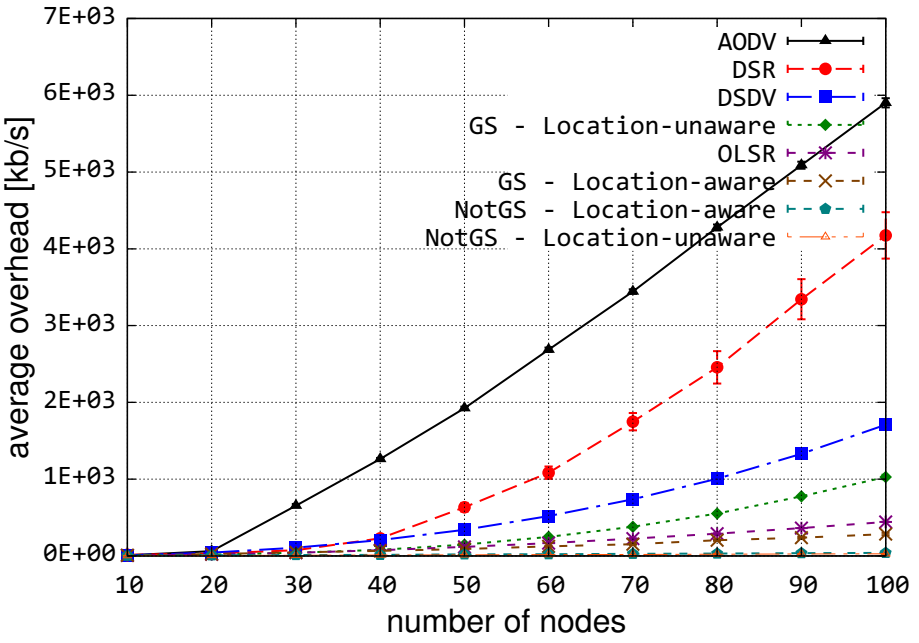

Figure B.6. Node density vs overhead (AODV, TDMA, RWP, 1200 m/s)

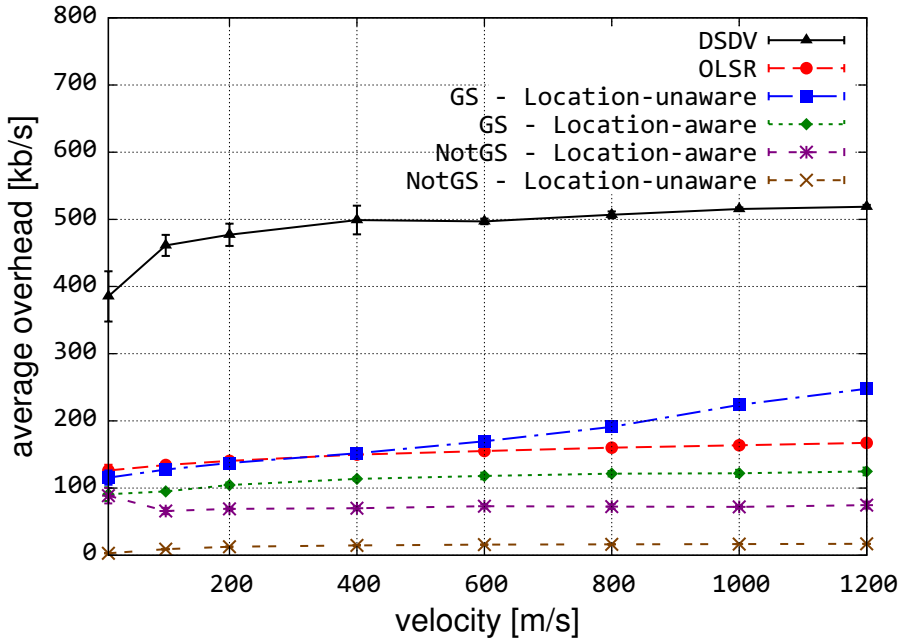

Figure B.7. Node velocity vs overhead (TDMA, RWP, 60 nodes)

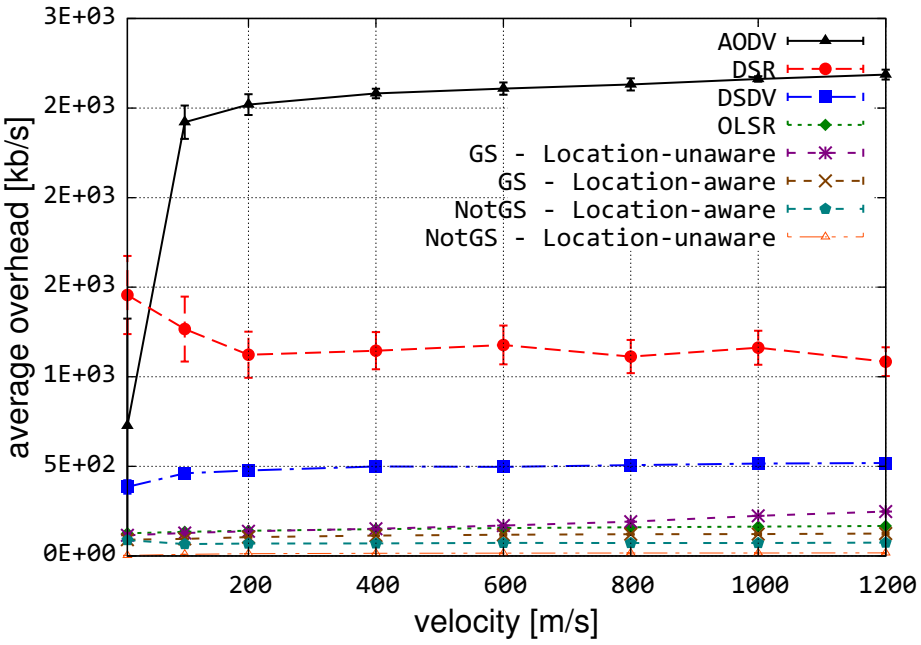

Figure B.8. Node velocity vs overhead (AODV, TDMA, RWP, 60 nodes)

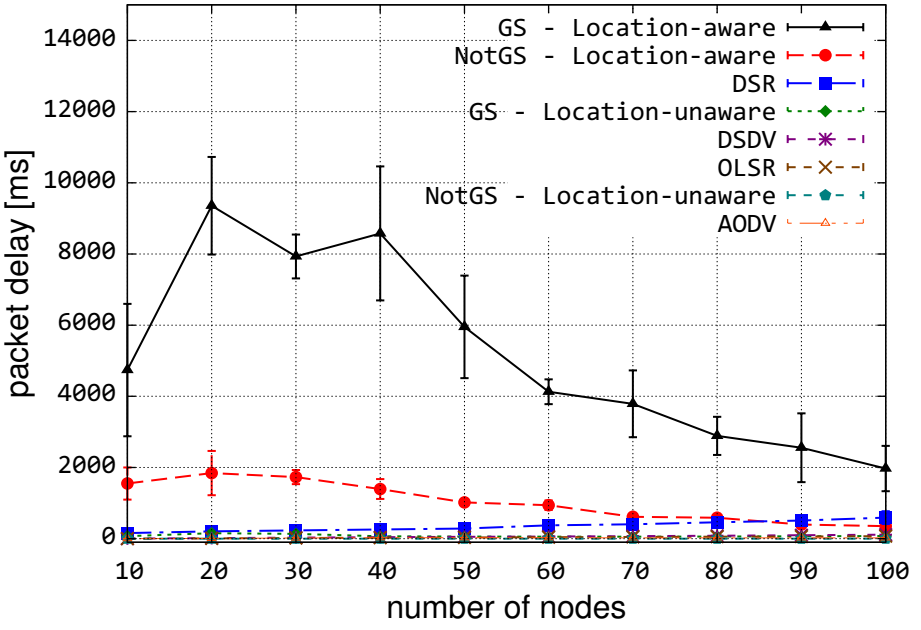

Figure B.9. Node density vs delay (TDMA, RWP, 1200 m/s)

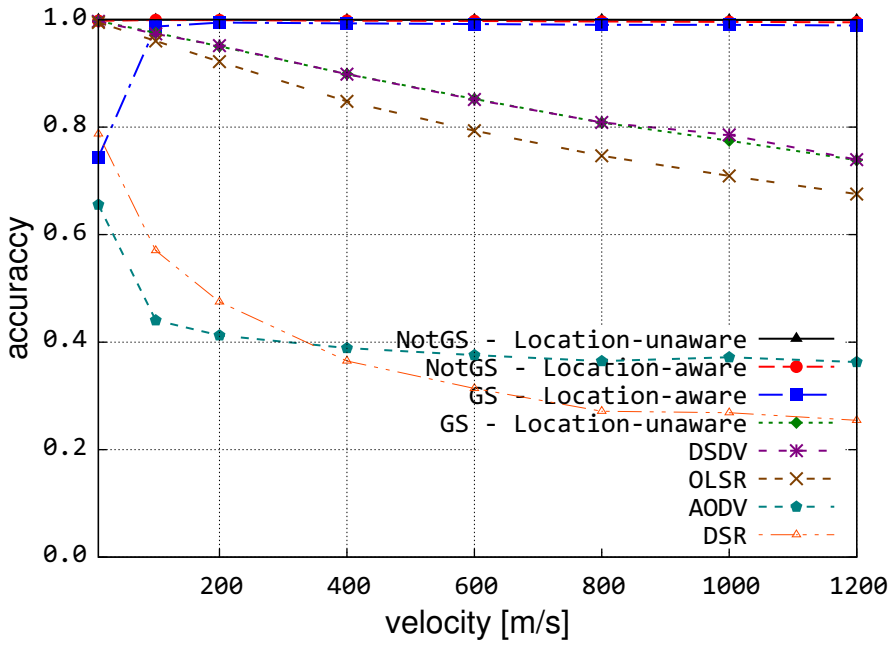

Figure B.10. Node velocity vs accuracy (TDMA, RWP, 60 nodes)

References

[1] Justin P. Rohrer, Abdul Jabbar, Erik Perrins, and James P. G. Sterbenz. Cross-layer architectural framework for highly-mobile multihop airborne telemetry networks. In *Proceedings of the IEEE Military Communications Conference (MILCOM)*, pages 1–9, San Diego, CA, November 2008.

[2] M. Mauve, A. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. *Network, IEEE*, 15(6):30–39, 2001.

[3] Stefano Basagni, Imrich Chlamtac, Violet R. Syrotiuk, and Barry A. Woodward. A distance routing effect algorithm for mobility (DREAM). In *Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, MobiCom '98, pages 76–84, New York, NY, USA, 1998. ACM.

[4] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (LAR) in mobile ad hoc networks. *Wirel. Netw.*, 6:307–321, July 2000.

[5] Egemen K. Çetinkaya and James P. G. Sterbenz. Aeronautical Gateways: Supporting TCP/IP-based Devices and Applications over Modern Telemetry Networks. In *International Telemetering Conference*, Las Vegas, October 2009.

[6] Kevin Peters. Design and Performance Analysis of a Geographic Routing Protocol for Highly Dynamic MANETs. Master's thesis, The University of Kansas, Lawrence, KS, June 2010.

[7] Imrich Chlamtac, Marco Conti, and Jennifer J. N. Liu. Mobile ad hoc networking: imperatives and challenges. *Ad Hoc Networks*, 1(1):13 – 64, 2003.

[8] M. Conti and S. Giordano. Multihop Ad Hoc Networking: The Theory. *IEEE Communications Magazine*, 45(4):78–86, April 2007.

[9] iNET Working Group. http://www.inetprogram.org.

[10] iNET Needs Discernment Report, version 1.0. Central Test and Evaluation Investment Program, May 2004.

[11] Justin P. Rohrer, Abdul Jabbar, Egemen K. Çetinkaya, Erik Perrins, and James P.G. Sterbenz. Highly-Dynamic Cross-Layered Aeronautical Network Architecture. *IEEE Trans. Aerosp. Electron. Syst.*, 47(4), October 2011.

[12] Abdul Jabbar and James P. G. Sterbenz. AeroRP: A Geolocation Assisted Aeronautical Routing Protocol for Highly Dynamic Telemetry Environments. In *International Telemetering Conference*, Las Vegas, NV, October 2009.

[13] Kevin Peters, Abdul Jabbar, Egemen K. Çetinkaya, and James P.G. Sterbenz. A Geographical Routing Protocol for Highly-Dynamic Aeronautical Networks. In *IEEE WCNC*, Cancun, Mexico, March 2011.

[14] Kevin Peters, Egemen K. Çetinkaya, and James P. G. Sterbenz. Analysis of a Geolocation-Assisted Routing Protocol for Airborne Telemetry Networks. In *International Telemetering Conference*, San Diego, CA, October 2010.

[15] D. B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proceedings of the 1994 First Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 158–163, Washington, DC, USA, 1994. IEEE Computer Society.

[16] Elizabeth M. Royer and C.-K. Toh. A review of current routing protocols for ad-hoc mobile wireless networks. *IEEE Personal Communications*, 6:46–55, 1999.

[17] S. Ramanathan and Martha Steenstrup. A survey of routing techniques for mobile communications networks. *Mob. Netw. Appl.*, 1:89–104, October 1996.

[18] Fan Bai, Narayanan Sadagopan, and Ahmed Helmy. Important: A framework to systematically analyze the impact of mobility on performance of routing protocols for adhoc networks. 2:825–835, March 2003.

[19] Xiaoyan Hong, Taek Jin Kwon, Mario Gerla, Daniel Lihui Gu, and Guangyu Pei. A mobility framework for ad hoc wireless networks. In *Proceedings of the Second International Conference on Mobile Data Management*, MDM '01, pages 185–196, London, UK, 2001. Springer-Verlag.

[20] Charles E. Perkins and Pravin Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *SIGCOMM '94: Proceedings of the Conference on Communications Architectures, Protocols and Applications*, pages 234–244, New York, NY, USA, 1994. ACM.

[21] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol (OLSR). RFC 3626 (Experimental), October 2003.

[22] C.E. Perkins and E.M. Royer. Ad-hoc On-demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, February 1999.

[23] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561 (Experimental), July 2003.

[24] D. Johnson, Y. Hu, and D. Maltz. The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4. RFC 4728 (Experimental), February 2007.

[25] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In Tomasz Imielinski and Henry F. Korth, editors, *Mobile Computing*, volume 353 of *The Kluwer International Series in Engineering and Computer Science*, chapter 5, pages 153–181. Kluwer Academic Publishers, Norwood, MA, 1996.

[26] I. Stojmenovic. Position-based routing in ad hoc networks. *Communications Magazine, IEEE*, 40(7):128 –134, July 2002.

[27] Silvia Giordano, Ivan Stojmenovic, and Ljubica Blazevic. Position based routing algorithms for ad hoc networks: A taxonomy. In *Ad Hoc Wireless Networking*, pages 103–136. Kluwer, 2001.

[28] H. Takagi and L. Kleinrock. Optimal transmission ranges for randomly distributed packet radio terminals. *Communications, IEEE Transactions on*, 32(3):246–257, March 1984.

[29] Ting-Chao Hou and Victor Li. Transmission range control in multihop packet radio networks. *Communications, IEEE Transactions on*, 34(1):38–44, January 1986.

[30] Justin P. Rohrer, Erik Perrins, and James P. G. Sterbenz. End-to-End Disruption-Tolerant Transport Protocol Issues and Design for Airborne Telemetry Networks. In *International Telemetering Conference*, San Diego, CA, October 2008.

[31]

[32] Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. MANET Simulation Studies: The Incredibles. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(4):50–61, October 2005.

[33] The network simulator: ns-2. http://www.isi.edu/nsnam/ns/, December 2007.

[34] The ns-3 network simulator. http://www.nsnam.org, July 2009.

[35] E. Weingartner, H. vom Lehn, and K. Wehrle. A Performance Comparison of Recent Network Simulators. In *IEEE International Conference on Communications (ICC)*, pages 1–5, June 2009.

[36] Hemanth Narra, Yufei Cheng, Egemen K. Çetinkaya, Justin P. Rohrer, and James P.G. Sterbenz. Destination-sequenced distance vector (DSDV) routing protocol implementation in ns-3. In *Proceedings of the ICST SIMUTools Workshop on ns-3 (WNS3)*, Barcelona, Spain, March 2011.

[37] Tyson Thedinger, Abdul Jabbar, and James P. G. Sterbenz. Store and haul with repeated controlled flooding. In *Second International IEEE Workshop*

on Mobile Computing and Networking Technologies (WMCNT), Moscow, Russia, October 2010.

[38] Josh Broch, David Maltz, David Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. *ACM MobiCom*, pages 85–97, Oct. 1998.

[39] Hemanth Narra, Egemen K. Çetinkaya, and James P. G. Sterbenz. Performance analysis of aerorp with ground station updates in highly-dynamic airborne telemetry networks. In *Proceedings of the International Telemetering Conference (ITC)*, Las Vegas, NV, October 2011. to appear.

[40] Wikipedia. Student's t-distribution. `http://en.wikipedia.org/wiki/Student's_t-distribution`, 2010. Online; accessed 17-May-2010.

[41] Dan Broyles, Abdul Jabbar, and James P. G. Sterbenz. Design and analysis of a 3–D gauss-markov mobility model for highly-dynamic airborne networks. In *Proceedings of the International Telemetering Conference (ITC)*, San Diego, CA, October 2010.

[42] Dousse O, Thiran P, and Hasler M. Connectivity in ad-hoc and hybrid networks. 2:1079–1088, November 2002.

[43] Aodv's rerr issue in ns-3. `https://www.nsnam.org/bugzilla/show_bug.cgi?id=1099`, July 2009.