Suitability Analysis of Routing Protocols in Vehicular Ad Hoc Networks for Distributed Multimedia Applications

by

Alan Byrne

Dissertation

Presented to Trinity College Dublin

in fulfillment of the requirements for the Degree of

Master of Science in Computer Science

Trinity College Dublin

Coláiste na Tríonóide, Baile Átha Cliath

September 2008

Declaration

I, the undersigned, declare that the work described in this dissertation is, except where otherwise stated, entirely my own work and has not been submitted as an exercise for a degree at this or any other university.

Alan Byrne

September 8, 2008

"Problems worthy of attack prove their worth by fighting back."

Paul Erdos (1913-1996)

Abstract

In this ever more connected world, the idea of vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication is relatively new. Research and development in the area is, nonetheless, highly active with groups or programs like the Vehicle Safety Consortium, the Car-2-Car Communication Consortium, the Advanced Safety Vehicle Program and the large-scale Vehicle Infrastructure Integration Program specifically set up to tackle issues related to V2V and V2I communication. This is due to the many applications for which these networks may be used, for example, early warning applications, automatic toll collection and multimedia applications. The vehicular ad hoc network (VANET) is a special application of V2V communication where many vehicles cooperate to provide an ad hoc network.

This dissertation explores the issue of the suitability of ad hoc routing protocols in VANETs for multimedia applications, specifically multi-player games. A critical review of current literature is performed to explore existing research in the field. It is discovered that routing protocols can be divided into position-based routing protocols and topology-based routing protocols. Position-based routing protocols may be better suited for application in VANETs due to their higher tolerance towards node mobility and their ability to scale.

Network simulations are used to analyse routing protocols in VANETs for their suitability towards the specific application of multi-player games. It is found that protocols which require the maintenance of specific routes for packet transmission fail to satisfy the requirements of packet delay and packet loss when the network topology is highly dynamic. Position-based routing protocols may, therefore, be better suited to supporting multi-player games in VANETs since they do not maintain routing paths through the network.

Position-based forwarding methods like greedy forwarding and contention-based forwarding, combined with a reactive location service, may achieve the necessary requirements of multi-player games in VANETs. Further research is required for this position to be fully confirmed. Multi-player games should be playable in VANETs sometime in the near future when VANETs become mainstream. However, additional research into routing protocols and other factors that affect the playability of multi-player games is first required.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

As research into Vehicular Ad Hoc Networks (VANETs) advances, it is possible that such networks could become more commonplace. One possible use of these networks is in multimedia communications. An example of such a use is children in cars playing multi-player games with children in other cars. This sort of application introduces new challenges to the wireless research community and reinforces old ones. As will be discussed later in this chapter, some categories of multi-player games require very high rates of packet delivery and tolerate only small latencies. These are not necessarily easy requirements to meet in VANETs. The highly mobile behaviour of VANETs makes wireless connections less dependable and necessitates research into routing protocols specifically designed for such highly dynamic environments in an attempt to meet the requirements necessary in, for example, multi-player games. This dissertation provides such a suitability analysis of routing protocols in VANETs for distributed multimedia applications.

The remainder of this chapter provides a brief overview of the network requirements for different categories of multi-player games. This provides the core information that must be kept in mind when assessing the potential use and suitability of routing protocols in VANETs towards multi-player games. The chapter then finishes by summarising the objectives and goals of this project.

1.1 Network Requirements for Multi-player Games

Multi-player distributed video games have become increasingly popular as a greater number of people have access to the Internet. A report by Parks Associates suggests that 34% of Internet users in the United States play games on the Internet on a weekly basis [1]. Since these games are being played across multiple machines, it is important to investigate the impact of latency, packet loss, bandwidth and connection duration on the state consistency and playability of real-time games to ensure that all players have the same view of the world and the gameplay is sufficiently smooth.

There are several different categories of multi-player games and based upon their characteristics, each has different network requirements. *First Person Shooters* (FPS) and *Massively Multi-player On-line Role-Playing Games* (MMORPG) are the most popular type of multi-player games followed by *Real Time Strategy* (RTS) games [2]. In FPS games, players view the world through the perspective of the character they play and move through the world shooting other entities in the game, which can include those controlled by other human players, with an assortment of weapons. Such games include Counter-Strike$^{\text{TM}}$and Halo$^{\text{TM}}$. MMORPGs allow players to interact with other players in different ways and support some form of character advancement. A popular MMORPG would be World of Warcraft$^{\text{TM}}$. In RTS games, players issue commands or orders to an assortment of units to accomplish different tasks but don't actually control any individual unit directly. Examples of such games are Command and Conquer$^{\text{TM}}$and StarCraft$^{\text{TM}}$. There are other categories of multi-player games, like racing games, but these are not discussed here.

1.1.1 First Person Shooters

The main user interaction components in FPS games are movement and shooting [3]. Movement shows significant resilience to both packet loss and latency. With latencies over 100ms shooting accuracy begins to degrade and position information becomes unsynchronised across clients resulting in one player's view losing synchronisation with the current game state, i.e. that player may be shooting at an object which has moved but the player's view hasn't been updated with the object's current location; the player is shooting at empty space but thinks they are shooting at an object. Racing games show similar network requirements with latencies beyond 100ms causing performance to degrade [4]. Artefacts, such as missing animations when firing a gun, become noticeable as packet loss approaches 3%. A general requirement for FPS games is that latency should not exceed 150ms and packet loss should not exceed 3% [3]. Bandwidth requirements tend to scale with the number of players and are bursty - packets are transmitted in short bursts as opposed to continuous uniform transmission, with the server hosting the game requiring significantly more bandwidth than the connecting clients. The bandwidth per player is generally fairly low, in the range of 40kbps for some games, to allow the game to be playable on the widest range of network speeds [5]. Average packet size is also fairly small, generally less than 200 bytes with an average closer to 100 bytes [5]. Players' session times for FPS games may follow a Wiebull distribution [6] with more than 60% of sessions lasting less than fifteen minutes and approximately 80% of sessions lasting less than thirty minutes.

1.1.2 Massively Multi-player On-line Role-Playing Games

MMORPGs can involve thousands of players connected to the same virtual world simultaneously. Similar to FPS games, packets are sent in bursts and the average bandwidth requirement is low, as low as 7kbps for some games [7]. A reason for the very low bandwidth could be the slow nature of MMORPGs. Packet sizes are also small with most less than 200 bytes in size and an average size of just over 100 bytes. The average connection time for a user in an MMORPG can be around 100 minutes for some games [7], much greater than that for a FPS. Some MMORPGs are able to run smoothly at latencies of up to a second [8].

1.1.3 Real Time Strategy Games

RTS games tend to support a small number of players in any given game, typically less than eight. They are not overly sensitive to latency, tolerating latencies of up to 500ms very well with minimal impact on gameplay [9, 10]. As with FPS games, bandwidth requirements are again kept sufficiently low to allow the game to be played on a modem connection.

1.1.4 Multi-player Games in VANETS

Table 1.1 shows the approximate breakdown of network requirements for the different categories of multi-player games discussed. With respect to VANETs, FPS games are probably the most suited form of game because of their lower session durations. Connection time is likely to be a significant constraint on multi-player games in VANETs. FPS games also happen to have the most stringent latency times, however, and this introduces a technical challenge to see if there exists any ad hoc routing strategies capable of providing these networking requirements.

1.2 Objectives

This dissertation investigates existing categories of routing protocols designed for ad hoc networks and tries to determine their suitability for application in VANETs, specifically

Table 1.1: Network Requirements for Different Categories of Multi-player Games

Game Type	Average Session Time	Tolerable Latency	Bandwidth Requirements
FPS	Short (60% of sessions last less than 15 minutes)	$< 150ms$	$\approx 40kbps$
MMORPG	Long (greater than 90 minutes)	Up to $1000ms$	As low as $7kbps$ but can be higher
RTS	Variable	Up to $500ms$	$\approx 40kbps$

towards their use in supporting multi-player games. This is essential due to the particular properties of VANETs, such as the potential of the communicating nodes to be moving at high velocities and the possibility of the networks to be of a very large scale. Ad hoc routing protocols which excel in static or slowly changing topologies may not be suited to the highly dynamic and large scale topologies of VANETs. Different routing protocols must be explored in order to determine which routing protocol or category of routing protocols perform best with respect to their tolerance towards highly dynamic topologies and scalability. The research for this dissertation is carried out through critical exploration of existing literature and by running network simulations.

The remainder of this dissertation is organised as follows. Chapter 2 explores existing or proposed routing protocols and analyses their suitability for application in VANETs based upon previous research. Chapter 3 describes the direction that this project took and the simulations that were performed in order to assess certain routing protocols for their suitability towards the specific application of multi-player games in VANETS and discusses the results. Chapter 4 presents the conclusions which can be drawn from this work and suggests potential future work which could be carried out.

Chapter 2

Analysis of Routing Protocols for Vehicular Ad Hoc Networks

Vehicular ad hoc networks (VANETs) are garnering increasing interest for their uses in safety applications, automatic toll collection, multimedia applications and more. VANETs have some unique properties such as nodes being able to move at high speeds resulting in very dynamic topologies as well as their ability to scale to very large sizes. It is important, therefore, to investigate routing protocols for VANETs to ensure efficient communication. Most ad hoc routing protocols fall into one of two categories: *topology-based* protocols and *position-based* protocols [11, 12].

Topology-based routing protocols use information about existing links in the network to make routing decisions when forwarding a packet from a source node to a destination node. They can be further sub-divided into proactive, reactive and hybrid approaches. A proactive protocol maintains route information to other nodes in the network independently of network traffic. A reactive protocol only maintains active routes or routes to nodes that are in use. A hybrid protocol tries to combine both approaches by, for example, combining a local proactive scheme with a global reactive one.

Position-based routing protocols require that the participating nodes be able to determine their physical position. This is typically achieved through the use of GPS. The routing decision at each node is then based upon the position of the destination and the position of the forwarding nodes' neighbours.

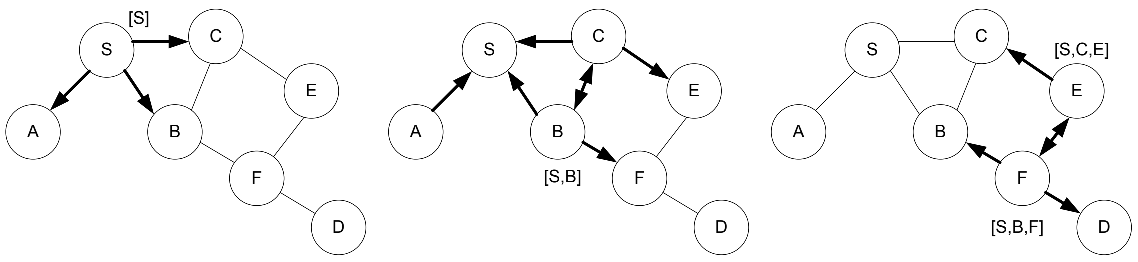
This chapter first provides two examples of standard reactive topology-based routing protocols and then an example of a proactive topology-based routing protocol. It then suggests why position-based protocols may be more suited to VANETs given their characteristic highly-dynamic topology and surveys the components of position based routing.

2.1 Topology-based Routing

As previously discussed, topology-based routing protocols can be defined as reactive, proactive or hybrid. This section describes Dynamic Source Routing and Ad hoc On-Demand Distance Vector Routing, two reactive protocols, and Optimized Link State Routing, a proactive protocol, as examples of topology-based protocols.

2.1.1 Reactive Protocol - Dynamic Source Routing

Dynamic Source Routing (DSR) [13] is a standard reactive ad hoc routing protocol. In DSR, when node S in figure 2.1 wants to transmit packets to node D but does not know any paths to node D, it initiates a route discovery. To do this, node S floods the network with Route Request (RREQ) packets. As the RREQ packet is propagated throughout the network, each node appends its own identifier to the packet resulting in the each packet containing the path that it travelled. In the example given in figure 2.1, S firsts appends its own identifier to the RREQ packet and broadcasts it to its neighbours A, B and C as shown in figure 2.1a. These nodes then append their own identifiers and rebroadcast the packet to their neighbours. In the case of node B, it transmits the RREQ packet containing the route [S,B] back to node S and also to nodes C and F as shown in figure 2.1b. S and C ignore the packet as they have already sent or received it. Finally, node D receives the RREQ packet from node F containing the route [S,B,F] as shown in figure



(a) Step 1 (b) Step 2 (c) Step 3

Figure 2.1: DSR - Propagation of RREQ

2.1c. Node D does not forward the packet since it is the intended target.

There are several things to note about the flooding of RREQ packets throughout a network. First of all, if node E was not connected to node F but was connected to node D, both nodes E and F could have tried to forward their packet to node D at approximately the same time. This could lead to the possibility of the packets colliding, increasing overhead as packets need to be retransmitted. One possible solution to this problem is to insert a random delay before forwarding the RREQ. Secondly, if the network was a lot bigger than shown in the example, it is possible that the RREQ packet would continue to be propagated throughout the network even after node D received it, wasting bandwidth.

After D receives the first RREQ packet, it responds to S with a Route Reply (RREP) packet. The RREP packet is sent to S along the route obtained by reversing the route contained in the RREQ packet. In the example given, the RREP packet travels along the path [D,F,B,S]. The RREP packet contains the route from S to D. Alternatively, if the links in the network are uni-directional, Node D can flood a RREQ packet to S with the RREP packet contained inside the RREQ packet. When node S receives the RREP packet, it caches the route to D. Subsequent data packets to D are routed by including the route to D in the header of the packet. Intermediate nodes use this route to decide whether they should propagate the packet or not. The data is “Source Routed”. If S tries to route a packet to D but a link breaks (identified through the lack of an acknowledgement for a packet, for example) and, say, node B can no longer communicate with F due to node mobility or any other reason, node B will return a Route Error (RERR) packet back to the original sender, in this case S. S and any node involved in forwarding the RERR packet to S update their routing tables to purge the broken route.

Another aspect of DSR is its use of route caching. Each node stores new routes in its routing table that it learns by any means. In the example given, when node S learns the route to D, it also learns the route to F. Node E learns a route to S when it receives the RREQ packet. Node B learns a route to D when it receives the RREP packet. When node B forwards a data packet, it learns a route to D. Route caching has three main advantages. Firstly, if a route from a source to a destination breaks, the source can use another route stored in its cache if one exists and avoid initiating a new route discovery. The other two advantages are that it can speed up route discovery and reduce the propagation of RREQs. This is depicted in figure 2.2. Node A wants to transmit to node D but does not know a valid route so it initiates a route discovery. However, node S has a cached route to D. Consequently, S can respond to A immediately with a route to D, greatly reducing the route discovery time. S does not forward the RREQ packet since it chose to respond with a RREP packet and since S is the only node within communication range of A, the flooding of the RREQ packet throughout the network is effectively reduced to zero, saving a potentially large amount of bandwidth. In reality, the reduction is generally not quite so significant. It is worth noting that route caching can also adversely affect performance. If a route breaks, a node may try several other stale routes obtained from its cache before finding a valid one. This wastes both time and bandwidth. There are techniques to try to mitigate this problem such as timing out inactive routes.

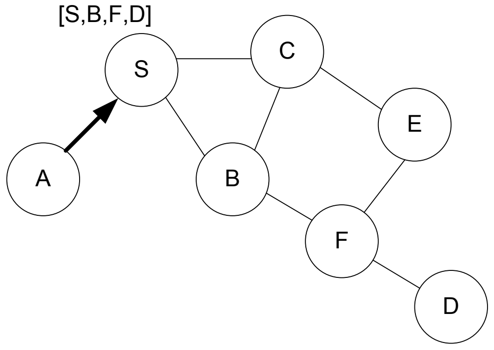


Figure 2.2: DSR - Advantages of Route Caching

Critique of DSR

Advantages

- Routes are only maintained between nodes that need to communicate. This reduces the size of the routing table at any given node.
- Route caching can reduce the overhead of route discovery.
- DSR has increased reliability as a single route discovery can return many different routes due to multiple intermediary nodes returning routes from their local caches.
- DSR is capable of operating using asymmetric links.

Disadvantages

- Packet header size is proportional to the number of hops on the route. This could be a considerable percentage overhead if transmitting a large number of small packets.
- RREQ packets could potentially reach all hosts in a network wasting a considerable amount of bandwidth.
- Since RREQ packets are being propagated by many nodes at approximately the same time, there is a danger of packet collision.
- An intermediary node may send a RREP packet containing a stale route which can adversely affect performance.

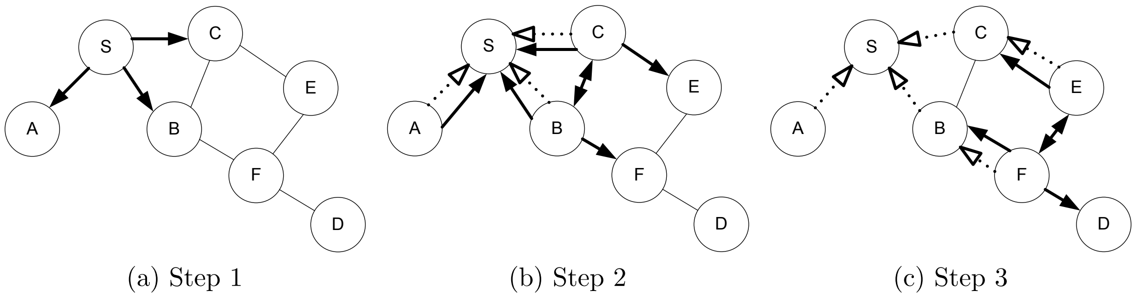
2.1.2 Reactive Protocol - Ad hoc On-Demand Distance Vector Routing

Ad hoc On-Demand Distance Vector Routing (AODV) [14] is another reactive ad hoc routing protocol. In DSR, a packets route through a network is inserted into that packets header. This means that a packet's header size is proportional to the number of hops along its route which can degrade performance when a packet's overall data size is small.

AODV attempts to improve on DSR by maintaining routing tables at nodes such that the packets no longer have to contain the transmission path.

In AODV, RREQ packets are propagated in the same manner as in DSR, however, instead of appending the route travelled to the header of a packet, when a node rebroadcasts a RREQ packet, it sets up a reverse path pointing towards the source as shown in figure 2.3 where node S is transmitting a RREQ packet to find a path to node D. The solid bold arrows represent the forwarding of a RREQ packet and the dotted arrows represent the setting up of a reverse path. In the example given, S first creates a RREQ packet and broadcasts it to its neighbours A, B and C as shown in figure 2.1a. The RREQ packet contains the source and intended destination node, sequence numbers associated with these nodes and a hop count. The nodes that receive the RREQ packet set up a pointer to the node that transmitted it (node S) and rebroadcast the packet to their neighbours as shown in figure 2.3b. Nodes S, B and C ignore the newly received packet as they have already sent or received it. When node D receives the RREQ packet from node F, as shown in figure 2.3c, node D sends a RREP packet which travels along the reverse path which has been set up between nodes D and S. The intermediary nodes B and F, as well as node S, set up a forward link when they receive the RREP packet. That is, node S will configure its routing table such that it knows that it must forward all packets destined for node D to node B. These routing tables are then used to forward data packets by including the destination address in the header of all packets.

When the initial RREQ packet is being transmitted, an intermediate node may re-



(a) Step 1 (b) Step 2 (c) Step 3

Figure 2.3: AODV - Propagation of RREQ

spond with a RREP packet if it knows a more recent path to the destination than the requesting node. A path is known to be more recent by examining the sequence numbers in the RREQ packet. Reverse paths in nodes' routing tables are purged after a specified timeout period. This timeout period must be long enough to allow a RREP packet to come back. Forward paths are also purged after a certain duration of inactivity. This is to ensure that only active routes are maintained. When a node detects that a link has broken, by, for example, failing to receive a HELLO message which all nodes must periodically broadcast, a RERR packet is transmitted to all neighbouring nodes that are routing through that link. The RERR packet purges the broken link from routing tables and updates destination sequence numbers. When a node is unable to transmit a packet due to a link breakage, a RERR packet is routed back to the original sender of that packet in order to update all routing tables along that path with respect to the link breakage.

Critique of AODV

Advantages

- Only active routes are maintained. This reduces the overall size of the routing tables.
- The use of routing tables at all nodes reduces the overhead of individual packets since the packet does not need to contain the entire route that it must traverse as is the case with protocols like DSR.

Disadvantages

- Unlike protocols like DSR which can cache multiple routes to a single destination, AODV may at most maintain one next-hop per destination.
- Routes will expire if they are unused even if the topology does not change. A node may have to perform a RREQ operation for a valid route which just expired.
- RREQ packets could potentially reach all hosts in a network wasting a considerable amount of bandwidth.

2.1.3 Proactive Protocol - Optimized Link State Routing

Optimized Link State Routing (OLSR) [15] is a proactive ad hoc routing protocol. It is an optimisation of standard link-state routing. Link-state routing tries to maintain the complete state of the network topology at each node in the network. This is accomplished by periodically flooding the status of network links throughout the network. All nodes broadcast the state of their current links and all nodes propagate link-state information sent by their neighbours. It is the responsibility of each node to keep track of the link-state information received from other nodes. Nodes then use local information to determine the next hop when forwarding packets.

OLSR introduces the idea of multipoint relays to link-state routing to reduce the overhead of flooding link-state information. While all nodes receive topology information sent by other nodes, only multipoint relays are allowed to retransmit this information. The multipoint relays for a node A are its direct neighbours such that all two-hop neighbours of A are a direct neighbour of at least one multipoint relay of A. Nodes periodically broadcast their one-hop neighbour list to enable nodes to select multipoint relays based upon their two-hop neighbours. A node is selected to act as a multipoint relay by its one-hop neighbours. Figure 2.4 shows an example of this. The nodes marked as MPR are the multipoint relays of node A. A possible strategy for selecting multipoint relays is to select direct neighbours that have the highest degree of connections to two-hop neighbours.

Critique of OLSR

Advantages

- The use of multipoint relays in OLSR can significantly reduce MAC contention and also the overall flooding of control packets when compared to standard link-state routing, reducing bandwidth usage.

- Since there is no specific “finding” operation required to set up a route to a destination, there is a low single-packet transmission latency.

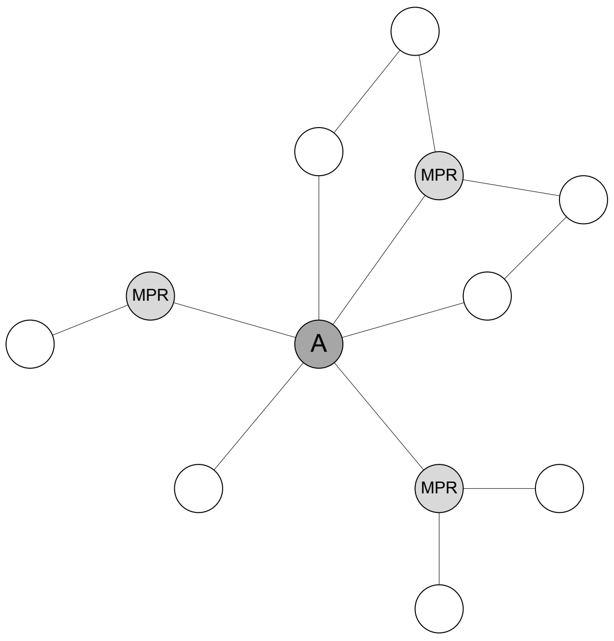


Figure 2.4: OLSR - Multipoint Relays

Disadvantages

- Bandwidth and storage space are used to maintain routes to all nodes which may include unused or inactive routes.
- If the network is sparse and all nodes are multipoint relays, the algorithm reduces to a standard link-state routing protocol.

2.2 Position-based Routing

Position-based routing protocols show advantages over purely topology-based protocols with respect to adaptivity and scalability [16]. Topology-based approaches need to, at least, maintain the network paths currently in use, limiting the amount of topology changes that can be tolerated within a given amount of time [11]. As previously stated, the routing decision in position-based routing protocols is based on the destination's position and the position of the forwarding node's neighbours. It does not require the establishment or maintenance of routes and can therefore be more suited to large networks with very dynamic topologies. The nodes in VANETs can move very fast resulting in significant and continuous topology changes. VANETs can also scale to rather large sizes. As such, position-based routing protocols may be a better choice over topology-based routing protocols for such networks [16].

Position-based routing consists of two components: a *location service* and a *forwarding strategy*. The *location service* is a method of discovering the physical location of the destination node for a given packet. The *forwarding strategy* is the method of actually propagating a packet through the network. The rest of this chapter discusses examples of proposed location services and forwarding strategies for position-based routing.

2.3 Location Services

Location services are used by nodes in an ad hoc network to determine the position of a destination node. They can be classified by the number of nodes in a network that host the service, either all or some, and the number of nodes for which a particular location server holds position information, either all or some. This results in four categories: all-for-all, all-for-some, some-for-all and some-for-some location services [11].

2.3.1 Distance Routing Effect Algorithm for Mobility - DREAM

Distance Routing Effect Algorithm for Mobility (DREAM) [17] uses an all-for-all location service. All nodes maintain a *location table* that contains position information for all other nodes believed to be in the network. Every node in the network is responsible for regularly flooding its position information to other nodes in the network. To reduce the level of flooding, the algorithm takes advantage of what the authors call the *distance effect*. The distance effect is the observation that the further away two nodes are from each other, the slower they appear to be moving with respect to each other. What is meant by this is that the greater the distance between two nodes, the slower the angle between them changes with movement. This is highlighted in figure 2.5. Node C is much further away from node A than node B is. If nodes B and C start at their positions at the top of the

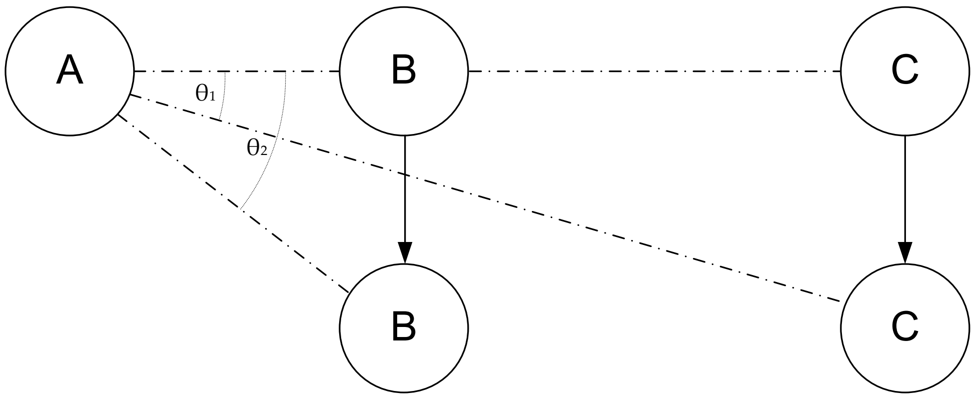


Figure 2.5: DREAM - Distance Effect

figure and both move the same distance to their positions at the bottom of the figure, the change of angle between node A and C is much less than that between nodes A and B.

To exploit the distance effect, control packets containing a node's position are assigned a time-to-live. The majority of control packets are only allowed to travel a short distance through the network. Control packets to distant areas of the network are broadcast less frequently since distant nodes do not need such accurate position information for routing purposes. The broadcast rate of position updates is also related to a node's mobility. The slower a node is moving, the less frequently it needs to flood position updates.

Evaluation of DREAM

DREAM also includes a specification for a forwarding strategy and as such a more detailed evaluation of the algorithm occurs in section 2.4.3 after the forwarding strategy has been discussed. Qualitatively, the algorithm seems very resistant to node failure as it tries to replicate position information for every node at every other node. Also, position queries only require a local lookup and as such are very fast. The algorithm may not be that scalable, however, due to the potentially large amount of flooding required for position updates, which may make the algorithm unsuited to large networks.

2.3.2 Quorum-based Location Service

The quorum-based location service [18] is typically a some-for-some location service. Some nodes in the network host position databases and each of these nodes only needs to have position information for some nodes in the network. In the quorum-based location service a subset of nodes in the network are dynamically chosen to act as a *virtual backbone* based upon the communication environment and network node density. Nodes in this virtual backbone communicate using a non-position based routing protocol. Each node in the virtual backbone holds a position database that contains the positions of some other nodes in the network. The nodes in the virtual backbone are divided into quorums or groups with a non-empty intersection. If, for example, there were six nodes in the virtual backbone, each numbered one to six, one possible set of quorums would be {1,2,3}, {1,4,5}, {2,4,6} and {3,5,6}. Each quorum shares at least one node with every other quorum.

When a node wants to update its position in the position database, it transmits its geographical position to the nearest backbone node. One method of identifying the nearest backbone node is to proactively maintain the positions of nodes within a local neighbourhood. Ideally, every non-backbone node is within direct communication distance of at least one backbone node. The backbone node then selects a quorum at random and distributes the position information to that quorum. The position information is time-stamped. When a node wishes to query another node's position it also contacts its nearest backbone node. The backbone node then collects position information from a quorum, usually different to the original quorum chosen to store the information, and returns this to the requesting node. Since the quorums have non-empty intersections, at least one of the nodes from the chosen quorum should have up-to-date position information. The requesting node identifies the most recent information from the time-stamps.

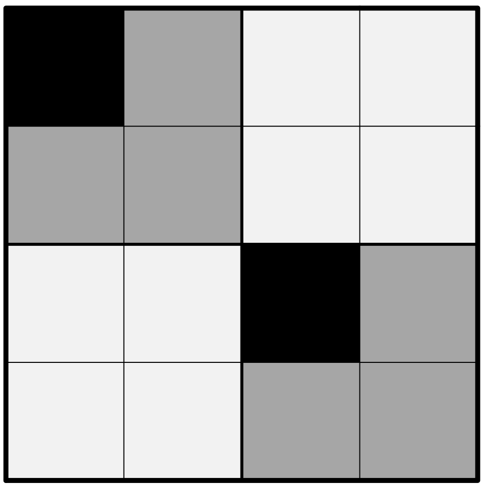
While the quorum-based location service is generally a some-for-some approach, since the quorum sizes can be adjusted, it can also be configured as an all-for-all or all-for some approach. The larger the quorum sets, the higher the cost of position updates and queries but the intersection of the quorums is also larger. This increases the resilience of the algorithm against unreachable backbone nodes. Generally, a balance needs to be struck between the cost of position updates and resilience against node failure or disconnection.

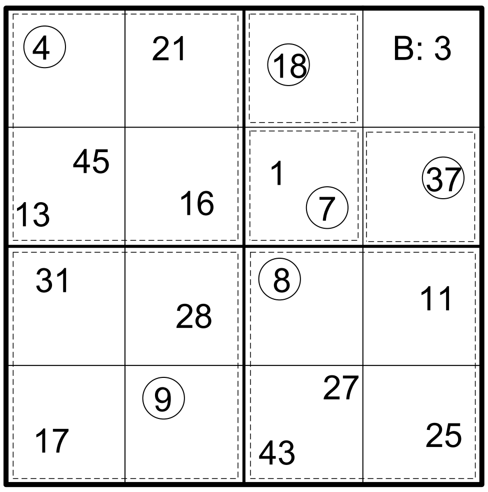
Evaluation of the Quorum-based Location Service

The quorum-based location service is rather flexible in that the size of the quorums can be adjusted to obtain the preferred trade-off between position update and query complexity and robustness. The resistance of the algorithm to node failure can, therefore, be adjusted to suit the network. The reliance of the algorithm on a non-position-based routing protocol may hurt its scalability.

2.3.3 Grid Location Service - GLS

Grid Location Service (GLS) [19] is an all-for-some approach. All nodes host position information for some other nodes in the network. In GLS the network area is divided up into a hierarchy of squares. Figure 2.6a shows a possible hierarchy. The smallest squares are referred to as order-1 squares. Four order-1 squares constitute an order-2 square, four order-2 squares constitute and order-3 square and so on. Any given order-n square is


(a) Grid Hierarchy



(b) Selection of Location Servers

Figure 2.6: GLS - Grid Hierarchy and Location Servers

only part of one order-$(n+1)$ square. For example, the four centre order-1 squares in figure 2.6a do not constitute an order-2 square. In this algorithm, all nodes must know the global partitioning of the grid.

All nodes in an order-1 square maintain position information about every other node in that order-1 square. This is achieved by localising periodic position broadcasts to the area of the square. Each node then uses *location servers* to distribute location information throughout the rest of the network. Every node acts as a location server for some other nodes. A node selects one location server in each of the surrounding order-2 squares and then one location server in each of the surrounding order-3 squares and so on. The density of location servers for a given node, therefore, decreases logarithmically with distance. Each node is hashed to an identifier. Taking node B in figure 2.6b as an example, it hashes to 3. It will then seek one location server in each of the inset squares. The location server will be the node with the identifier closest to but higher than 3 in each of the squares. A modulus numbering system is used so that if there are no nodes with a higher identifier it starts again counting from zero. The eventual location servers for B are those circled. A node does not need to know the identifiers of its position servers. Position information can be geographically forwarded to the square that is to contain the location server. The first node inside that square to receive it can then begin the position update process by routing the request to nodes with closer identifiers to the original node performing the position update.

Figure 2.7 shows a node A making a position query for node B. The numbers above each node represent the nodes that the current node knows about by virtue of being one of their location servers. Node A requests B's position from a node it knows about with an identifier closer to but greater than or equal to B's identifier. The query then gets repeatedly forwarded using the same process until it reaches B which then responds to A with its current position.

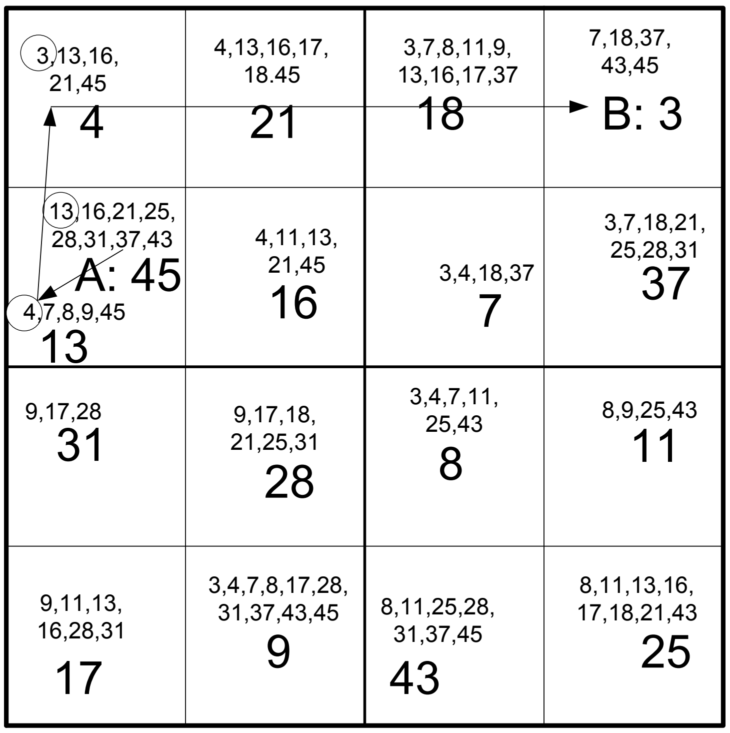


Figure 2.7: GLS - Querying a Position

Evaluation of GLS

GLS is simulated in [19]. A network density of approximately 100 nodes per square kilometre is used and an order-1 square is set at 250m by 250m. The number of nodes is varied between 100 and 600. The physical area of the network is adjusted to maintain the average node density. A random waypoint mobility model is used [20]. In this model a node randomly selects a point in the network area and moves to it at a constant speed. When it gets there it chooses a new random point and a new speed and begins moving towards it after a set pause time. This process continually repeats. In these simulations, the speed of nodes is limited to between 0m/s and 10m/s and no pause time is used.

The query success rate for position information decreases with increasing number of nodes but always remains well above 80%. Query success rate is also affected by node speed. As nodes move faster, the success rate begins to drop. It is shown that if the nodes are not uniformly distributed throughout the grid, it is possible that a few nodes may end up acting as location servers for a disproportionately large number of other nodes and require large amounts of space to store this information. The effect of node failure is tested for a network of 100 nodes. Queries are only ever generated to nodes that are active. The query success rate degrades very gracefully as nodes are periodically turned off and on.

GLS is also simulated using a random waypoint mobility model in [21]. The order-1 squares are set as 250m by 250m and nodes have a maximum speed of 10m/s. For the first scenario, a network of size 2km by 2km with 400 nodes is used. About 7.1% of queries fail. The main reasons for this are that the request traveled too far or entered a routing loop (time-to-live expired) or no node with a closer identifier to that of the destination could be found.

Next, the number of nodes is varied between 100 and 400. The number of failed queries increases significantly as the number of nodes is reduced to 100 but the main reason for these failures is no route existing between two nodes when using a greedy forwarding strategy (see section 2.4.1) to transport packets. In most cases there existed some route to the destination. As the number of nodes increases, the query failures due to a packet exceeding its time-to-live also increase. The authors in [21] show that node mobility can actually cause looping of packets to occur in GLS as nodes cross grid boundaries and their location servers are not updated in time.

2.3.4 Reactive Location Service - RLS

The reactive location service (RLS) [22] is actually a one-for-one location service. RLS has similarities to the route request phase of DSR. When a node wants to communicate with another node but does not know its position, it issues a location query packet which is flooded through the network. The packet contains the position and identifier of the requesting node and the identifier of the destination node. The packet also contains a sequence number associated with the source and intended destination of the location query packet so nodes can identify if they have already forwarded the packet. The packet is forwarded through the network until it reaches its intended destination or a specified time-to-live (TTL) expires. If the destination is not reached, it is assumed to be unreachable but a retry still occurs since the factors governing an unreachable node, for example, network partition, are subject to change. There is a limit to the number of retries that are attempted.

The flooding of location query packets can also be performed as a “binary search”. That is, the packet is first flooded through the network with a low TTL. If the destination node is not found, the TTL for the packet is set at its maximum and the packet rebroadcast. This attempts to take advantage of situations where communication is mostly local.

On reception of a location query packet at the destination, a location reply packet is created which contains its position information and is transmitted back to the querying node. It is forwarded using the underlying routing protocol. Reception of the location reply packet at the querying node completes the position request process.

Evaluation of RLS

RLS is simulated with binary search using a greedy forwarding strategy in [22]. A random waypoint model is used for node mobility. The area is set as 2000m by 2000m with the number of nodes ranging from 100 to 400. The nodes’ speed is also varied between 10m/s and 50m/s. GLS with a greedy forwarding strategy and DSR are also simulated for reference.

The RLS scheme achieves delivery ratios in excess of 90% in all simulations except in the case of 100 nodes. In the case of 100 nodes, DSR shows a 20% better delivery ratio compared with RLS, which achieves a delivery ratio of 60-75%, but drops significantly with increasing number of nodes and increasing node speeds to as low as 14%. DSR struggles with scaling and increased node mobility. RLS shows no such problem whem combined with a greedy forwarding strategy. It tolerates highly dynamic topologies and appears to scale well. A reason for the poor performance of RLS in the 100 node scenarios could be due to the greedy forwarding strategy. Greedy forwarding can perform poorly in sparse networks as will be shown later.

GLS shows similar results to RLS except in the case of dense high-speed scenarios where GLS shows poorer packet delivery. GLS also shows slightly poorer results in the case of 100 nodes moving at 10m/s. A possible reason for GLS's poor performance in dense high speed scenarios could be that as a node's velocity increases, it sends more position update packets. This can result in network congestion if all nodes are doing this.

One-hop latency times for RLS are between 0.01 seconds and 0.5 seconds with the best performance occurring for 200 nodes moving at 30m/s. Network latency can only be kept small, however, if the network is not congested but the same is shown to be true for GLS and DSR as well. RLS is also simulated using an exponential search where the search radius is gradually increased in an exponential manner if a destination node is not located by adjusting the flooded packet's time-to-live. The overhead and one-hop latency gets uncontrollably high for this method for dense high-speed simulations. GLS shows similar results to RLS with binary searching for one-hop latency times except, once again, for the case of dense high-speed simulations where GLS approaches 2 seconds latency times.

2.4 Forwarding Strategies

Forwarding strategies in position-based routing are the algorithms used by nodes to transport a packet through the network after the source has already discovered the position of the destination. There are several different strategies.

2.4.1 Greedy Packet Forwarding

In greedy packet forwarding, the position of the destination is included in the header of all packets and nodes keep forwarding the packet in the general direction of the destination. As such, a local maximum (there exists no node within range of the forwarding node closer to the destination) can cause the method to fail even if a path does exist to the destination. A recovery algorithm can be used to get around this. A recovery algorithm tries to find a path to the destination so long as one exists but is less efficient than greedy packet forwarding. An example of such an algorithm is the face-2 algorithm [23]. It is unclear whether or not the costs of using such an algorithm are worth the benefit or if a simple timeout-based approach would be sufficient [16]. There are several different strategies for greedy packet forwarding.

MFR - Most Forward within Radius

In MFR, a packet is forwarded to the neighbouring node that the forwarding node knows about that makes the most progress towards the destination. This technique tries to minimise the numbers of hops that a packet must traverse in order to reach the destination. It is suitable if nodes cannot adjust their signal strength.

NFP - Nearest with Forward Progress

In NFP, a packet is forwarded to the nearest neighbour closer to the destination. This method can show better performance than MFR if nodes are able to adjust their signal strength [11]. Reducing the transmission range of all packets being forwarded can significantly reduce the probability of packet collision and increase the average packet progress.

Compass Routing

In compass routing, a packet is forwarded to the neighbour closest to the straight line drawn between the forwarding node and the destination node. This method tries to minimise the spatial distance that any given packet travels.

Randomly

It has also been suggested to forward a packet randomly such that the sender randomly chooses a node closer to the destination than itself to which to forward the packet. This method tries to minimise the accuracy of the information needed about neighbours' positions.

Evaluation of Greedy Packet Forwarding

Greedy Perimeter Stateless Routing (GPSR) is simulated in [24]. GPSR is a greedy forwarding strategy which uses the MFR strategy and also includes a recovery strategy if a packet reaches a local maximum. DSR is also simulated and the results compared. The random waypoint model is used for node mobility. Nodes pause for an interval after they reach a waypoint. Simulations are carried out for 50, 112 and 200 nodes. Nodes move at a maximum speed of 20m/s. The network area is adjusted to maintain an average of one node per 9000 metres squared. There are on average 20 neighbours within direct communications range of any node. In the simulations, when a node creates a new packet to be dispatched, it includes the true current location of the destination node.

Over 97% of packets are successfully delivered with GPSR for a network with 50 nodes which is just slightly higher than DSR. GPSR shows significantly less overhead when compared with DSR in terms of the number of packets being transmitted for the protocols to work. Of all delivered packets, GPSR delivered 97% of these along the most optimal path while DSR only managed this for 84.9% of packets. Increasing the diameter of the network while maintaining the density has little effect on the delivery rates of data for GPSR. DSR's delivery ratio, on the other hand, drops significantly with increasing network size. This is due to the fact that DSR needs to maintain full end-to-end routes.

A downside to greedy forwarding strategies is that a packet needs to contain the location of the destination within a one hop transmission range. Greedy forwarding strategies also show low delivery rates in sparse networks [25].

2.4.2 Contention-based Forwarding

Contention-based forwarding (CBF) [26] has similarities to greedy forwarding. CBF forwards packets to the neighbour closest to the destination but eliminates the need for a node's neighbours to periodically send beacons to distribute position information. This is done by allowing the forwarding node's neighbours to decide who should rebroadcast packets.

In CBF, a node firsts broadcasts a packet as a one-hop broadcast to its neighbours. The position of the forwarding node is included in the packet. All nodes which receive this packet then check to see if they are closer to the destination than the forwarding node. It they are, they start a timer inversely proportional to the amount of progress that the packet has made to the destination. A node that will make optimal progress will have a timer of zero seconds, where optimal is defined as the intersection of a circle defining the transmission range of the forwarding node and a line drawn from the forwarding node to the destination node. When the timer runs out, they rebroadcast the packet. A node does not rebroadcast the packet if they detect another node having already done so. The intent of this is that, hopefully, only one node will forward the packet and that node is the node that will make the most progress towards the destination. It is still possible, however, that more than one node could re-broadcast a packet, for example, two nodes which make approximately the same progress towards the destination. This process is repeated to forward packets through a network.

Since it is the node that makes the most progress towards the destination that forwards a packet, the delay that the timer introduces is kept, generally, very small. The algorithm also only uses the very accurate position information that a node has about itself in addition to the expected position of the destination node.

Evaluation of CBF

CBF is simulated in [26]. The results are compared with those for an optimised greedy forwarding strategy with a beacon period of one second which allows packets to be rerouted to a different node if the selected next hop has moved out of range before the transmission occurs. The random waypoint mobility model is used for an area of 2km by 2km with varying node densities. It is first shown that without mobility, there is a low packet delivery ratio for a network containing 100 and 200 nodes. This is due to the presence of local maxima in sparse networks. CBF actually shows a higher delivery ratio than greedy forwarding, though. This is because packet duplication in CBF can result in a packet being forwarded along a non-greedy path.

Tests with mobile nodes are then carried out with 300 nodes. For speeds of 10m/s, 30m/s and 50m/s, CBF shows packet delivery ratios of around 98%. The delivery ratio for the optimised greedy forwarding strategy is just slightly less. With 400 nodes, CBF delivers 100% of packets. CBF is also shown to use much less bandwidth than greedy forwarding with the main overhead for the greedy forwarding scheme coming from the beacons required to distribute location information among neighbours. The average hop latency is also tested for a network with 300 nodes. For speeds of around 10m/s, the greedy forwarding scheme shows a latency of less than 0.005s while the latency for CBF is approximately 0.01s. With increasing speed, however, the average latency for CBF slowly drops while it increases drastically for greedy forwarding, exceeding that for CBF at around 30m/s.

2.4.3 Restricted Directional Flooding

Restricted directional flooding has similarities to greedy packet forwarding in that nodes forward packets to other nodes physically closer to their destinations. However, in restricted directional flooding, the forwarding node forwards the packet to multiple other nodes. Basically, a packet is flooded through the network but the area that the packet is allowed to traverse is controlled.

DREAM

DREAM [17], discussed under location services, also defines a forwarding strategy. In DREAM, an *expected region* is defined as the area where the destination node is believed to reside. The region is shown in figure 2.8. The centre of the region is the last known position of the destination. The radius is defined as $(t_1 - t_0)v_{max}$, where t_1 is the current time, t_0 is the timestamp of the most recent position information that the source has about the destination and v_{max} is the maximum possible speed at which the destination may travel. Where S is the source and D is the destination, the packet is flooded to all

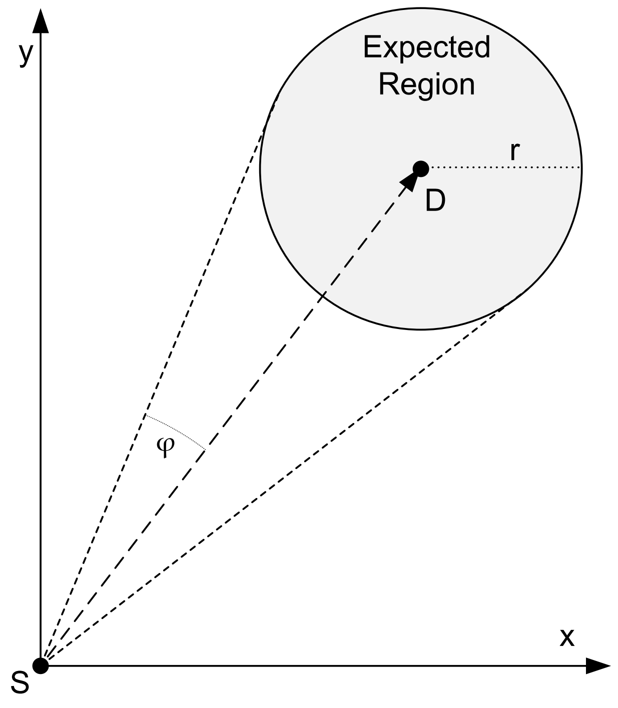


Figure 2.8: DREAM - Expected Region

nodes in the direction of the expected region. That is, within the the angle 2φ as shown in the diagram. Intermediary nodes then repeat this process of defining an expected region for the destination and forwarding the packet. It is expected that nodes closer to the destination will have more accurate location information about the destination. A recovery algorithm can be used if a packet cannot be forwarded to the destination in this manner but a specific recovery algorithm is not included in the protocol.

Location Aided Routing - LAR

Location Aided Routing (LAR) [27] is not a complete position-based routing protocol but a mechanism to limit the flooding of route requests in reactive ad hoc routing protocols. It defines an *expected region* containing the destination node in the same way as DREAM, using the node's last known position and maximum possible velocity. It then defines a *request zone* which is a rectangle that contains both the source node and the expected region. The request zone is explicitly included in route request packets, which are then flooded through the network. However, only nodes within the request zone may forward the packet. If the route request fails, the sender may start a new discovery procedure with a larger request zone, which may be the entire network. The rest of the route discovery protocol is similar to DSR as previously discussed.

Evaluation of Restricted Directional Flooding

The full DREAM algorithm is simulated in [17]. 30 nodes are placed on a grid of size 100 units by 100 units and given a speed of grid units per 100 simulation clock ticks. It is shown that greater than 80% of packets are delivered to the destination without the use of a recovery algorithm for varying node speed. The algorithm is then compared to DSR. DREAM is shown to have a lower end-to-end delay than DSR with DSR giving an end-to-end delay of between 25% and 250% larger than that of DREAM.

Restricted directional flooding algorithms are very robust in that they provide many different routes to the destination. They are also tolerant of position inaccuracy with respect to the destination's location since they flood an entire area in which the destination is believed to be as opposed to routing towards a particular point. They consume significant bandwidth in the process of flooding data through the network, however, and as such may not scale very well to large networks with high volume network traffic.

2.4.4 Hierarchical Routing

Another forwarding strategy is to form a hierarchy of routing protocols. This section gives examples of two such protocols. The methods that follow use a proactive non-position based protocol for local routing but switch to position-based routing for efficient long-distance routing of packets.

Terminodes Routing

Terminodes Routing [28] is comprised of the two routing strategies: Terminodes Local Routing (TLR) and Terminodes Remote Routing (TRR). TLR defines the use of a proactive non-position-based distance vector routing scheme for transmissions to local nodes defined as an area whose radius is given in terms of hops. TRR is a greedy position-based routing scheme used for transmitting packets long distances. To avoid local maximums during the long distance packet forwarding, the idea of Anchored Geodesic Packet Forwarding (AGPF) is introduced. Basically, the sender of a packet defines fixed geographical points that a packet must traverse on the way to the destination to avoid local maximums. The sender obtains the information to define these points through communication with nodes with which it is already in contact, for example, nodes within the local routing area. When a forwarding packet gets close to the destination, as in the destination is reachable using the TLR strategy, the forwarding strategy switches back to TLR.

Grid Routing

Grid Routing [29] is a similar approach to Terminodes Routing. It uses a non-position-based proactive distance vector routing scheme at the local level and a greedy position-based routing algorithm for long-distance routing. An additional component of grid routing is that it accommodates nodes that do not know their own position. In each area where the proactive distance vector routing protocol is being used, there must be at least one position-aware node. Position-unaware nodes may then use the position of the position-aware node as their own position. In this way, the position-aware nodes act as a proxy for the position-unaware nodes. Packets for the position-unaware nodes arrive at the position-aware nodes where they are then forwarded to the position-unaware nodes using the proactive distance vector routing scheme.

To avoid local maximums in the greedy routing scheme for long-distance routing, Grid Routing implements the idea of Intermediate Node Forwarding (INF). INF is best explained using an example. In figure 2.9 node S wants to send a packet to node D. Each node is only in direct communication range of its closest neighbours and there exists a route [S,A,B,C,E,D]. m is the midpoint of a line drawn between S and D. S first forwards the packet to A, which is a local maximum. A has no neighbours closer to D than itself. The packet is dropped and a error is reported back to the sender. The sender, S, then

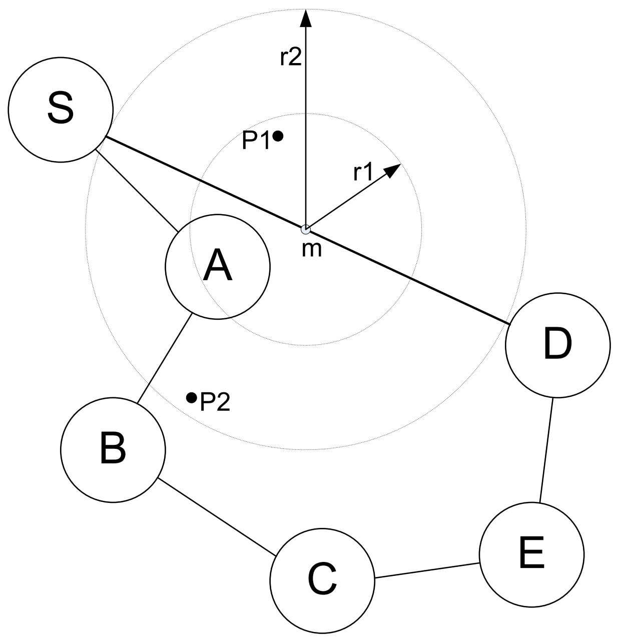


Figure 2.9: Grid Routing - Intermediate Node Forwarding

defines a disc of radius $r1$ centred at m, the midpoint. The radius is initially set as one quarter the distance between the source and destination nodes. The sender then chooses a random point inside the disc, say $P1$, that the retransmitted packet must traverse. A is the closest node to $P1$ and so the packet is dropped again and an error reported. The radius of the disc is then doubled to $r2$, a random point $P2$ is chosen and the packet is retransmitted. This time node B is closer to $P2$ than A so A forwards the packet to B. Since B is the closest node to $P2$, the forwarding strategy switches back to greedy forwarding and the packet is forwarded to D through C and E. In INF, the radius of the circle is continually increased until a path to the destination is found or until the destination is declared unreachable. It is possible that this strategy could fail to find a route to the destination even if one exists since it is effectively attempting to find a way around a local maximum by guessing but it is a very lightweight strategy in terms of complexity and state information required.

Evaluation of Hierarchical Routing

Terminodes Routing is simulated in [28]. The network area is 5400m by 1000m and contains 600 nodes. The random waypoint mobility model is used with a maximum speed of 20m/s and a pause time of 10 seconds. The protocol is first simulated with and without the TLR part. There is a better packet delivery rate when TLR is used. This is because TLR makes the algorithm more tolerant of position inaccuracies since a packet can arrive at any node within the TLR zone where the destination resides and the destination will receive the packet. As the location information that a source node has about a destination node becomes more accurate, the delivery rate for the algorithm without TLR begins to converge with that for the algorithm with TLR.

The algorithm is then tested with and without the AGPF component. A restricted random waypoint mobility model is used for these tests. The model is intended to reflect towns connected with highways. Four areas in the network are defined as towns. Nodes move randomly within a town for a while and then move to another town. A node may only move between towns that are "connected." This reflects the fact that a given town is not directly connected to every other town with a highway. It is assumed that nodes have access to a *map* reflecting the layout of the towns. It is used by nodes to define the anchored points for AGPF when routing packets between towns since a packet must move along the highways from town to town to reach a distant destination. The recovery strategy from GPSR is included in the greedy packet forwarding strategy. A network area of 3000m by 2500m is used with 500 nodes and a maximum speed of 20m/s. It is shown that up to 20% more packets are delivered when AGPF is used. The performance of the algorithm using AGPF falls as nodes move between towns more frequently but always shows a better packet delivery ratio than without it.

Grid Routing is simulated in [29]. A random waypoint model is used for node mobility. For 300 nodes in a 2km by 2km area, the location proxy system is tested. It is shown that the size of routing tables is only slightly increased when compared with the same protocol not using the location proxy system. The additional work required by nodes acting as a proxy is significant, particularly when there is a large percentage of nodes that do not have a location system. However, it is still shown that so long as the network is sufficiently dense, a network where only 10% of nodes know their position is feasible.

The INF system is shown to be ineffective in networks where nodes are randomly distributed. INF is also tested for a network where a node's movement is restricted to mimic that of an urban environment. Rectangular blocks are inserted into the network area which block both node movement and transmissions to simulate the idea of roads and buildings. In this scenario, INF shows higher delivery rates than for the protocol without INF (effectively greedy packet forwarding). The advantage obtained from using INF lessens as streets become wider. For streets less than 30m in width, INF still has a packet delivery rate of less than 80%.

2.5 Conclusions

VANETs can have a very dynamic topology and can easily scale to very large sizes. Position-based routing protocols may be a better choice compared to topology-based routing protocols for VANETs as they can scale well and may handle high network mobility better. Examples of both location services and forwarding strategies for position-based routing have been discussed.

Location services like GLS seem to be the most universally applicable. All nodes share responsibility for storing position information but each node will only store position information for some other nodes. It scales well. The volume of state to be stored at each node grows with $O(log(n))$ and the communication complexity for updating and querying position data grows with $O(\sqrt{n})$. However, the performance of GLS seems to fall in dense high speed networks.

RLS seems to cope well with node mobility. RLS is actually implemented and tested with a greedy forwarding strategy in [30]. For a static 3-hop scenario with the nodes positioned such that packets must be routed around an obstacle, a UDP throughput of 400kbit/s is shown to be achievable for packets of size 1444 bytes. The throughput depends on packet size. An increase in the number of packets leads to a higher probability of collisions. A throughput of 450kbit/s is achievable with TCP. A mobile 3-hop scenario is also performed with the nodes travelling around a 5km track for a load of 150kbit/s in the presence of other vehicles and traffic lights. Packet losses occur after a destination node moves out of communication range of a forwarding node but before the forwarding node begins to forward the packets through a different intermediary node. This situation occurs because a node only identifies if another node has moved out of communication range through a timeout mechanism. Poor link quality was also a notable factor for packet loss at times.

Greedy forwarding strategies show promise as a suitable form of packet forwarding for VANETs. They are a very efficient form of packet forwarding and suited to highly dynamic topologies. They forward packets along a near-optimal path if one exists. They also provide very high delivery rates for densely-populated networks but they can have low packet delivery rates in sparsely-populated networks. They also require rather accurate position information for the destination.

Contention-based forwarding, being similar to greedy forwarding, also shows promise. It eliminates the need for a node to maintain position information about its neighbours. Greedy forwarding and CBF are simulated and compared in [31]. The simulation consists of a 12km highway with nodes travelling between 50km/h and 220km/h with varying traffic densities. The simulation is carried out with varying distances between the source and destination nodes. A topology-based protocol is also simulated. The topology-based protocol shows the lowest packet delivery ratio. The position-based protocols are shown to be robust against node mobility with both CBF and greedy forwarding achieving delivery rates above 99.7%. CBF shows significantly less load in achieving this. The position-based approaches also show round trip times of less than 0.05 seconds for all distances. In low network densities, the delivery rate for the position-based approaches can drop to as low as 97% while there is a minimal impact on round trip time.

The effect of a reactive location service on the protocols is also examined. RLS causes a constant load increase for all distances although the load increase experienced by the greedy forwarding strategy is actually less than that for CBF. The round trip time for the first packet sent in a communication exchange also increases. The length of the additional delay is related to distance.

Hierarchical schemes are also interesting approaches for packet forwarding in VANETs. The schemes discussed here use a greedy forwarding strategy over long distances but the use of a local proactive non-position-based scheme reduces the required precision of position information when routing to distant nodes.

Chapter 3

Routing Protocol Simulations

This project was originally intended to continue on from a previous dissertation by Piyao Liu titled “Rear-seat Multiplayer Gaming using Peer-to-Peer Car Communication Networks” [32]. The previous work developed a routing protocol to discover nodes in a Vehicular Ad Hoc Network (VANET). The network comprised of Microsoft^{TM}Xboxs^{TM}connected to PDAs to provide wireless connectivity. The Ad hoc On-Demand Distance Vector (AODV) routing protocol, a reactive topology-based routing protocol, was then used to route messages between these nodes. The previous work also investigated the use of a GPS system to identify node location and direction which would be used to gauge the suitability of nodes in joining a game. The idea was, when selecting nodes to join a game, to choose nodes which would have a connection time sufficiently long for the game to complete. Neither the PDAs nor the GPS devices were available during the previous project and as such it was not possible to include them in the testing process.

The original purpose of this project was to fully implement Liu’s design, since the equipment had since become available and to investigate if the AODV routing protocol could be switched to a different protocol which may be more suited to the high speed environment of the VANET, perhaps taking into account the availability of the GPS information. Subsequent to the investigation of available routing protocols, position-based routing protocols were chosen for evaluation of their potential application in multi-player gaming across VANETs. There proved, however, to be insufficient implementations of such routing protocols to be able to apply and compare them in terms of latency, packet loss and throughput in a real environment. The direction of the project, therefore, had to change.

Most of the position-based routing protocols discussed in the relevant research had, at some point, been implemented on network simulators. The simulator implementation for one of these routing protocols, Greedy Perimeter Stateless Routing (GPSR), was available for the NS-2 simulator [33]. Few other simulator implementations were available. The goal of the project was redefined to be the simulation and comparison of GPSR, a position-based routing protocol, and AODV, a topology-based routing protocol using the NS-2 simulator with realistic vehicle movement patterns. It could then be shown whether or not position-based routing protocols such as GPSR were a potentially useful routing protocol when compared with topology-based routing protocols for the purpose of multimedia applications such as multi-player games across VANETs.

3.1 Configuration of Routing Protocols

Before simulations could be carried out, the network simulator NS-2 had to be installed and configured to use the two routing protocols GPSR and AODV. NS-2 runs in a Linux environment and, for simplicity, the Linux operating system Ubuntu^{TM}was set up to run in a virtual machine using VMware^{TM}Player running inside Microsoft^{TM}Windows XP. The installation of NS-2 required compiling the code and setting some environment variables. NS-2 version 2.33 was used. NS-2 comes with a working implementation of AODV. A NS-2 GPSR patch written by Ke Liu [34] was then applied to the NS-2 installation to allow simulation of the GPSR routing protocol. The patch was written for NS-2 version 2.26 or later. After attempting to apply the patch to the simulator, it was found that some of it, specifically the source code files that replaced existing files, was no longer compatible with the newer versions of NS-2. The patch files which were supposed to replace existing files were examined and compared to the original files to discover precisely what had to be changed for the patch to work. The NS-2 source code files were then modified to produce the equivalent result as what the GPSR patch required in order to get GPSR running. The implementation of GPSR used an idealised location service. Packets were annotated with the true location of the destination. As a consequence, any results obtained would represent the best possible outcome for GPSR. Further research would be needed to observe the effect of using a location service with GPSR, as would be required for its deployment in a real-world scenario.

3.2 Vehicle Movement Patterns

With the network simulator and routing protocols up and running, the next step was to acquire realistic vehicle movement patterns so that the routing protocols could be analysed and compared with each other for their suitability in VANETs. Next Generation Simulation (NGSIM) [35] is a program which develops algorithms in support of traffic simulations. It recently published vehicle trajectory data for segments of several different roads in the U.S.A. The information included position and velocity information of vehicles in the recorded areas over periods of fifteen minutes. These datasets were acquired for this project and a small program was written in C++ to convert these files into node-movement files suitable for use in the NS-2 network simulator. Deriving node-movement data for the simulations from actual vehicle trajectory data would provide greater confidence in any assertions made about the effectiveness of different routing protocols in VANETs from the simulations.

The length of the road from which the traces were taken was 640m. There were traffic lights located at four of the five intersections along the road. The approximate speed of the vehicles when they were moving was 35km/hr. 1222 vehicles were tracked over the fifteen minute period.

3.3 Traffic Patterns

In order to analyse the routing protocols for their effectiveness towards distributed multimedia applications, the traffic pattern produced by such applications needed to be simulated. The network requirements of on-line multi-player games was investigated for this purpose. There are many different categories of multi-player games and each has different network requirements. The network requirements for First Person Shooter (FPS) games were simulated as FPS games showed the lowest session durations of the different game categories investigated and session duration is likely to be a key constraint for multi-player games in VANETs. Vehicles participating in a game must remain sufficiently close to each other in order to provide continuous connectivity and satisfy latency requirements for the duration of the game.

Typically in a FPS game, multiple clients connect to a single server. As such, the testing in the NS-2 simulator involved setting multiple different nodes connecting to the one other node which acted as the server (client-server architecture). The bandwidth usage was then approximated in the network simulator using the research performed. A bandwidth of 25kbps was maintained each way on every link in the simulation for a total of 50kbps on every link. The size of each packet was set at 200 bytes and they were transmitted at an interval of 0.064 seconds to maintain the bandwidth usage. The traffic was simulated for periods of approximately 110 seconds. This was a limitation set by the time that nodes took to travel along the street.

3.4 Simulation Results

The simulations that were run tried to compare the two routing protocols, AODV and GPSR, with respect to throughput (bit rate), packet delay and packet loss. In the end, however, it was not possible to get any trace information for these three metrics for GPSR. The reasons for this are not entirely clear. The GPSR implementation was old and the method used to create a node capable of using the GPSR routing protocol in the

NS-2 simulator was also an old method. A different implementation of GPSR was then obtained, one which came already integrated into NS-2. However, the version of NS-2 that the protocol came with was nearly ten years old and a successful compilation of the source code was never achieved. It may have simply not been compatible with modern compilers. Eventually, time ran out. With the change in direction of the project and the use of an unfamiliar network simulator, NS-2 (a tool with a significant learning curve), which ran inside an unfamiliar operating system, Linux, too much time was consumed and there was insufficient time remaining to fix the problems which were occurring with the GPSR routing protocol.

In the end, only AODV was simulated. Multiple scenarios for AODV were simulated in order to properly assess the suitability of the protocol for application in VANETs for multi-player games. The radio range for each node during the simulations was set at 100m. There was no background traffic simulated in the network. The results for the different scenarios are now presented.

3.4.1 Two Nodes Close Together

AODV was first simulated for two nodes communicating with each other. The simulation lasted 110 seconds, from 140 seconds to 250 seconds simulation time. The nodes started off approximately 200m apart but traveled almost along side each other from the 180 second mark. Table 3.1 summarises the results for AODV in this scenario. Both average packet delay and the packet drop percentage were very low. Figure 3.1 maps the packet drop rate against simulation time and it can be seen from this that the majority of packets were dropped around the 160 second mark. The packets were dropped as the two nodes

Table 3.1: Simulation Results for "Two Nodes Close Together"

Packets Transmitted	Packets Dropped	Percent Dropped Packets	Packets Received	Packets With Delay Over 150ms	Average Packet Delay
3438	23	0.67%	3415	4	17.5ms

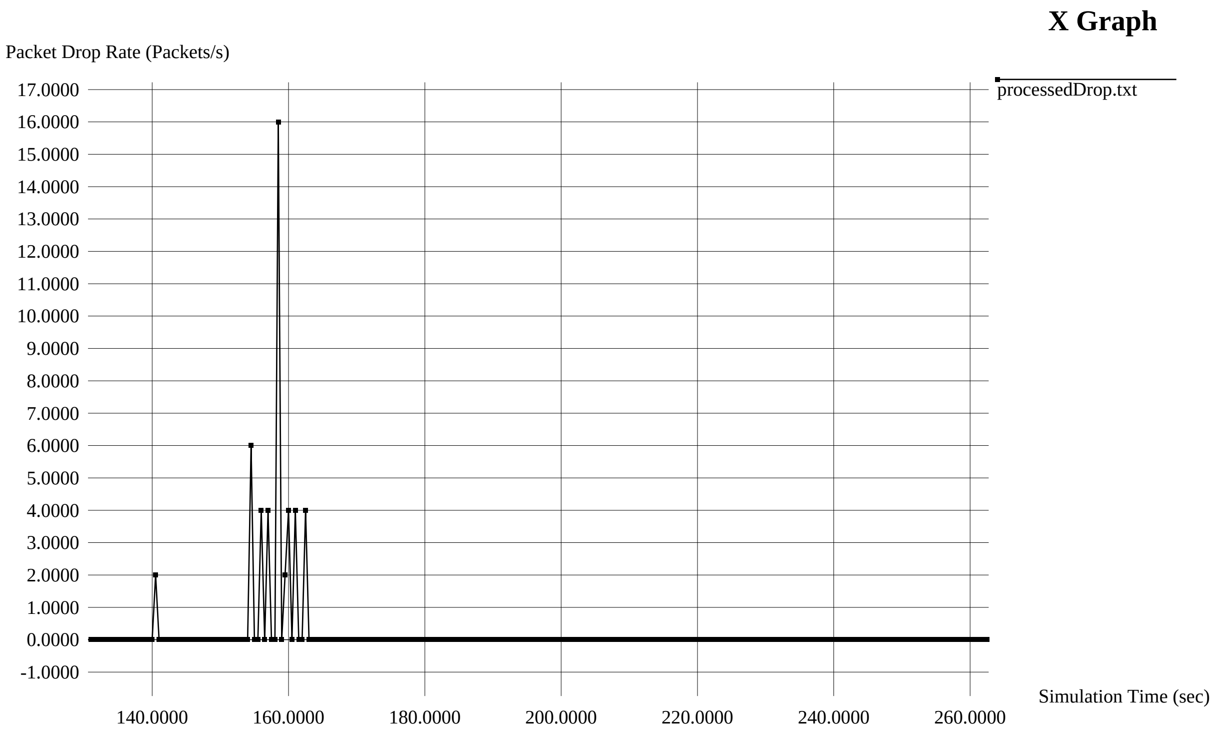


Figure 3.1: Packet Drop Rate for “Two Nodes Close Together”

moved closer together, that is, as the path between the two nodes changed. In this case, the path switched from multi-hop to single-hop. This scenario produced similar results for different seeds of the random number generator.

3.4.2 Two Nodes Moving in Opposite Directions

The next scenario examined communication between two nodes starting at opposite ends of the street, moving in opposite directions to each other. The simulation lasted 104 seconds, from 167 seconds to 271 seconds simulation time. Table 3.2 summarises the results for this scenario. The average packet delay and the packet drop percentage were very extreme. Figure 3.2 shows the throughput between the two nodes during the simulation.

Table 3.2: Simulation Results for “Two Nodes Moving in Opposite Directions”

Packets Transmitted	Packets Dropped	Percent Dropped Packets	Packets Received	Packets With Delay Over 150ms	Packets With Delay Over 600ms	Average Packet Delay
3252	1604	49.32%	1648	284	210	366.4ms

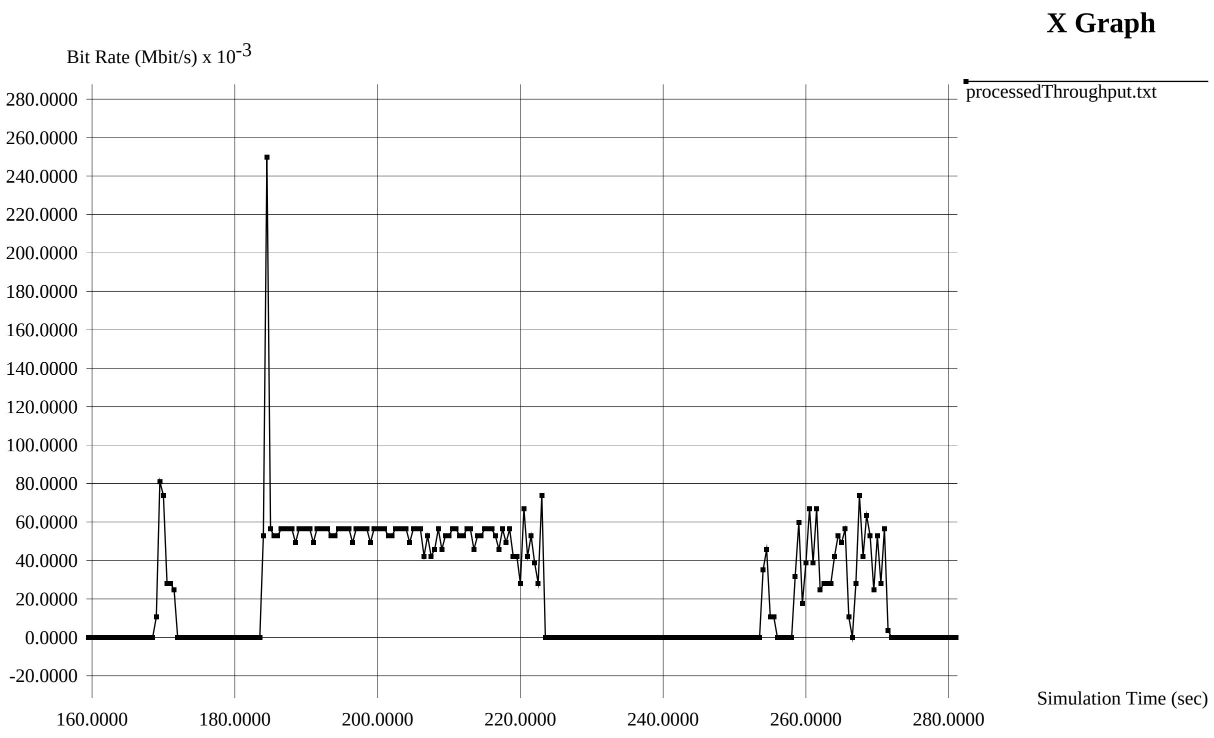


Figure 3.2: Throughput for "Two Nodes Moving in Opposite Directions"

Two significant periods can be seen for which there was no connection between the two nodes and the throughput fluctuates just before these periods. The final period of connectivity is also very unstable. The two nodes were approaching the opposite ends of the street at this time. Figure 3.3 shows the delay for each individual packet sent during the simulation. Packets with no corresponding delay are packets that were dropped. It can be seen from this graph that the packet delay for this unstable period is very varying but generally very large.

3.4.3 Five Nodes

AODV was simulated for five nodes communicating in a client-server architecture. The simulation lasted for 110 seconds, from 140 seconds to 250 seconds simulation time. The nodes were spread along the street moving in the same direction. The distance between the front and back nodes was approximately 250m but they clumped together due to traffic lights around the 190 second mark and from around the 230 second mark until the end of the simulation. Table 3.3 summarises the results obtained for this scenario. The

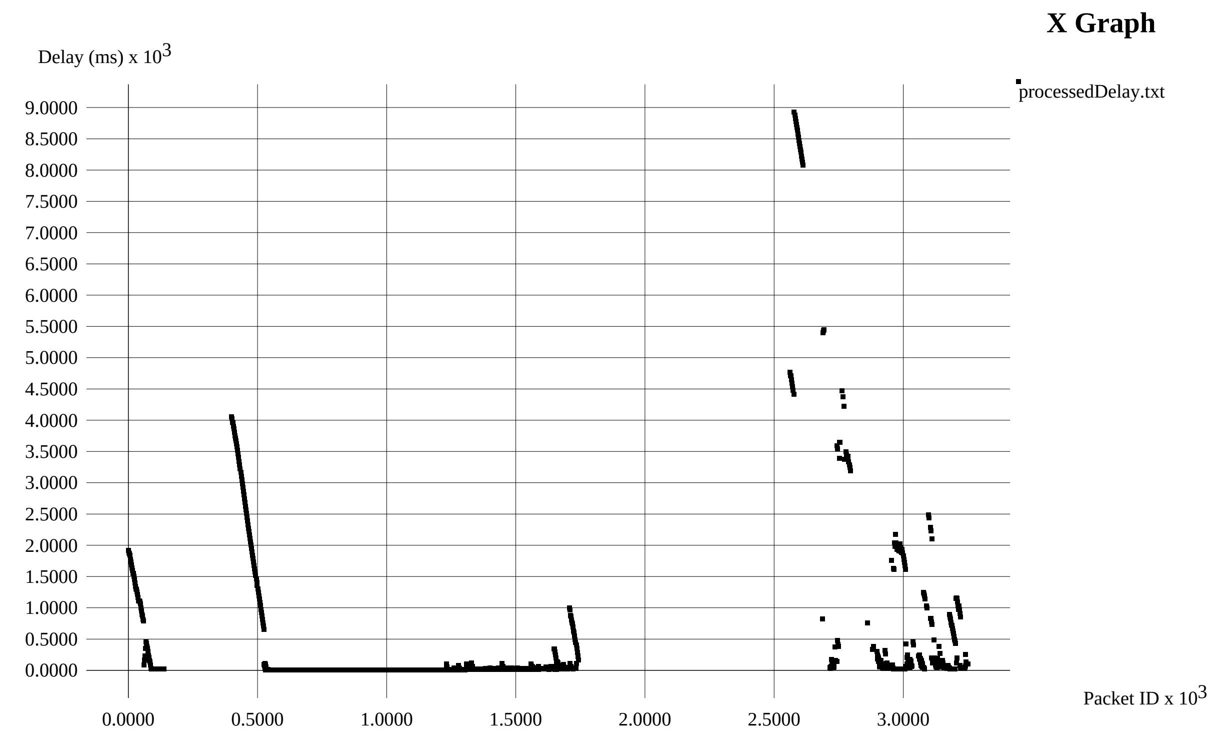


Figure 3.3: Packet Delay for "Two Nodes Moving in Opposite Directions"

Table 3.3: Simulation Results for "Five Nodes"

Packets Transmitted	Packets Dropped	Percent Dropped Packets	Packets Received	Packets With Delay Over 150ms	Average Packet Delay
13752	38	0.28%	13714	35	22ms

average packet delay and the packet drop percentage were very low. Figure 3.4 shows the packet delay for each individual packet sent during the simulaion. It shows that the delay for the vast majority of packets transmitted was quite close to the average packet delay. This scenario produced similar results for different seeds of the random number generator.

3.4.4 Six Nodes

The "six node" scenario was very similar to the "five node" scenario. Six nodes were communicating in a client server architecture for 110 seconds, from 140 seconds to 250 seconds simulation time. The nodes were distributed along a 250m stretch of road but

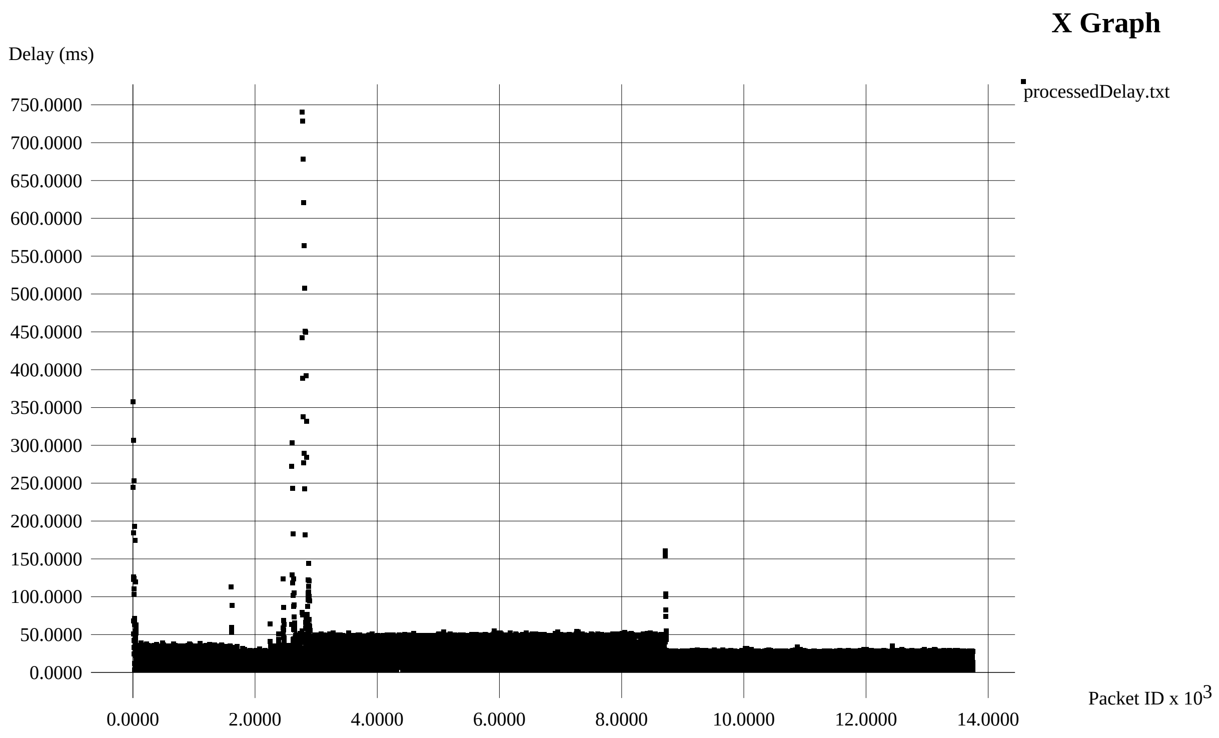


Figure 3.4: Packet Delay for "Five Nodes"

clumped together at around 190 seconds and from 230 seconds to the end of the simulation. This scenario produced drastically different results dependant on the seed for the random number generator. Two sets of simulations are presented in table 3.4. Simulation set 1 shows a very significant packet drop rate and a very high average packet delay while simulation set 2 shows a rather small packet drop rate as well as a short average packet delay. By comparing the graphs of throughput for simulation 1 and 2 in figures 3.5 and 3.6 respectively, it can be seen that it was approximately between 160 seconds simulation time and 215 seconds simulation time where the performance of AODV significantly diverged. Figure 3.7 shows that packets were being dropped throughout this period in simulation

Table 3.4: Simulation Results for "Six Nodes"

	Packets Transmitted	Packets Dropped	Percent Dropped Packets	Packets Received	Packets With Delay Over 150ms	Packets With Delay Over 600ms	Average Packet Delay
1	17190	1193	6.94%	15997	5150	4115	339.9ms
2	17190	110	0.63%	17080	931	56	48ms

1. Figure 3.8 shows the packet drop rate for simulation 2. Figures 3.9 and 3.10 show the distribution of the individual packet delay for simulation 1 and 2 respectively.

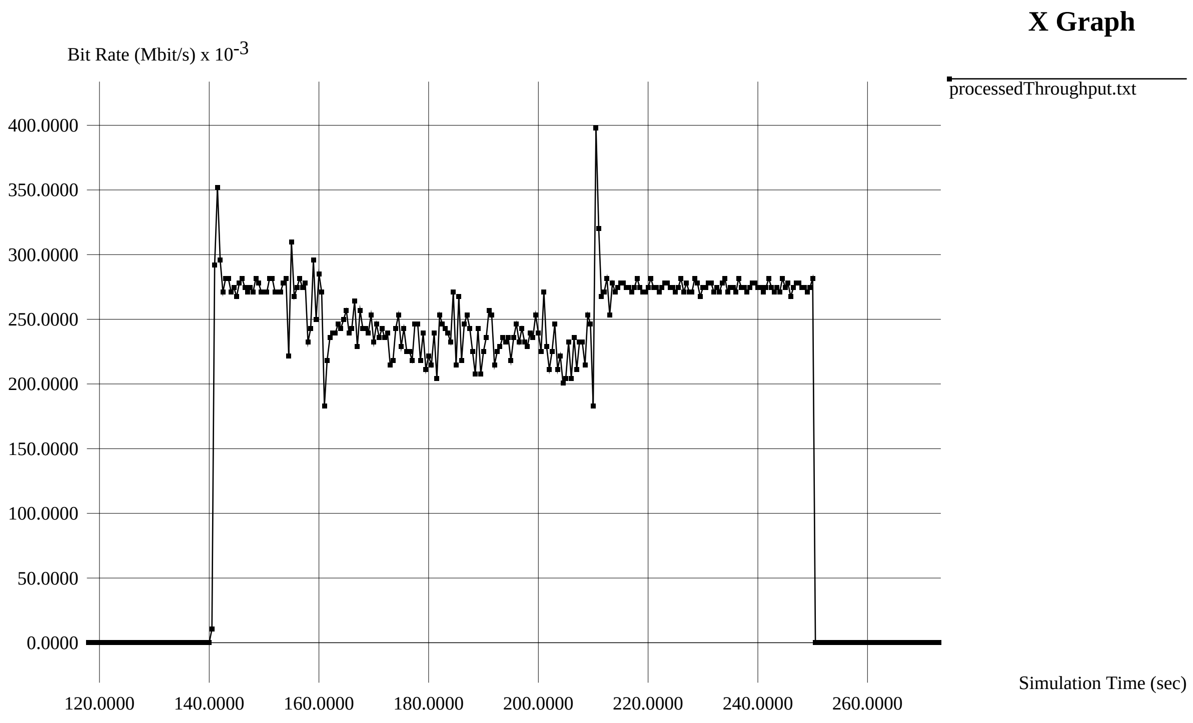


Figure 3.5: Throughput for “Six Nodes, Simulation 1”

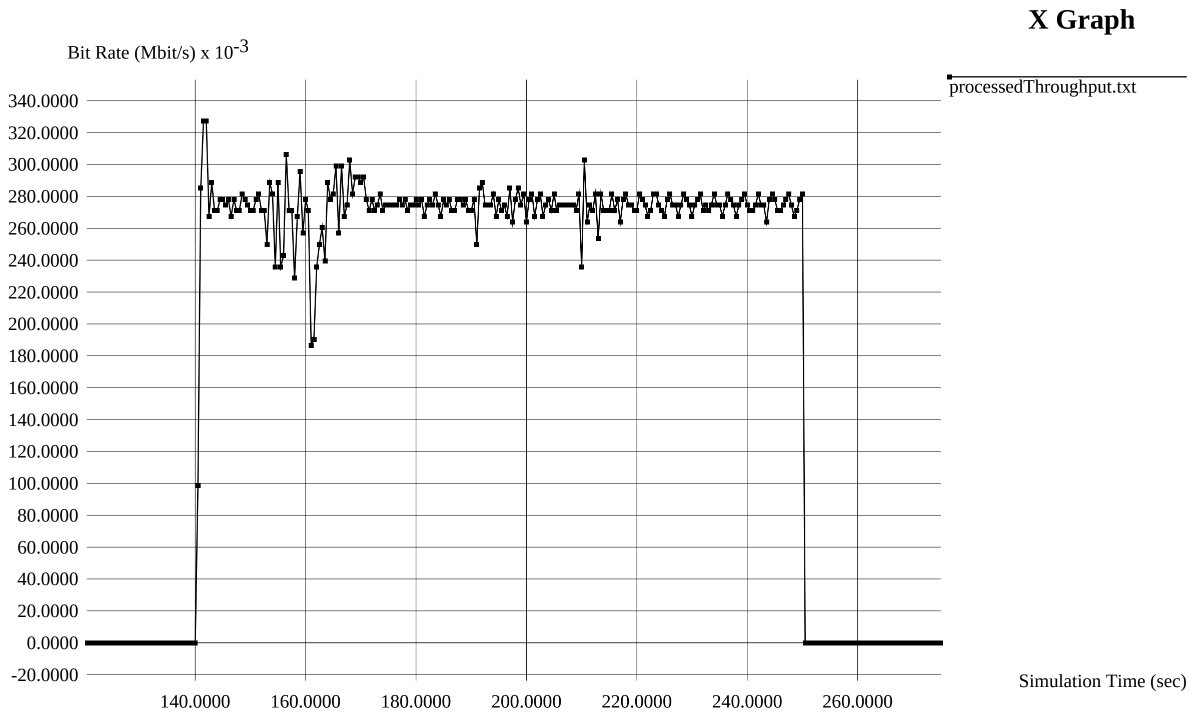


Figure 3.6: Throughput for “Six Nodes, Simulation 2”

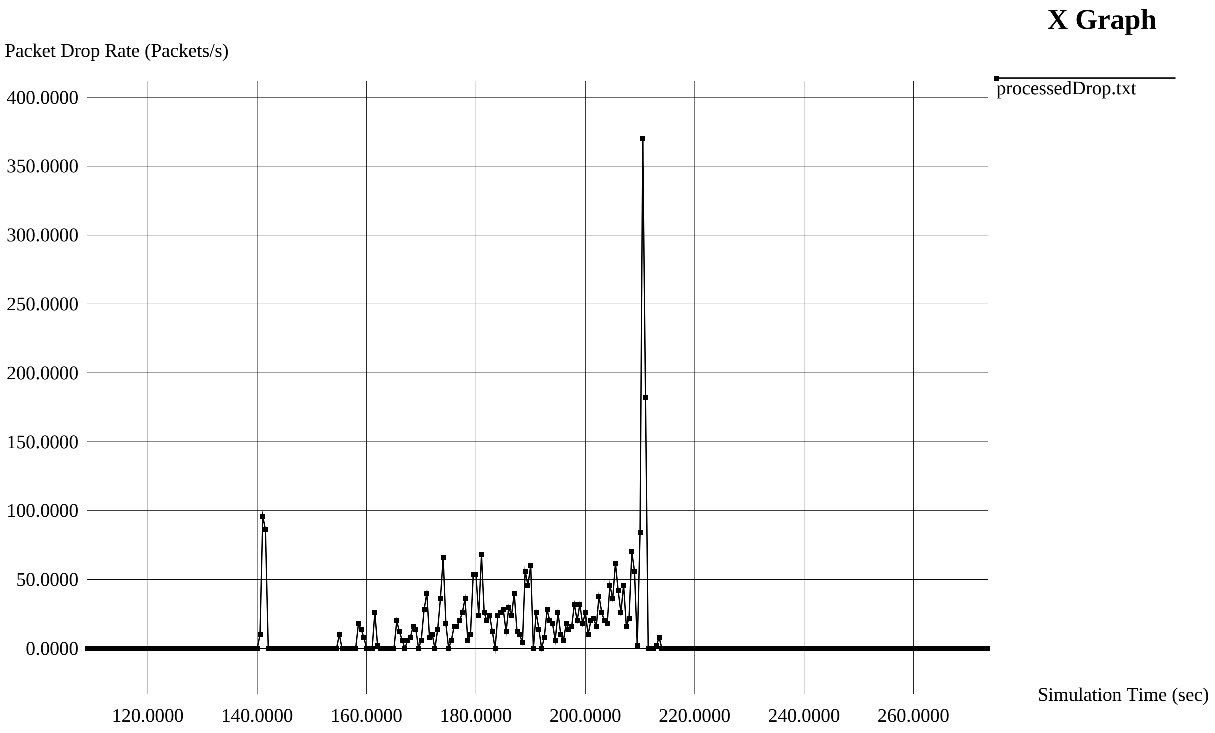


Figure 3.7: Packet Drop Rate for "Six Nodes, Simulation 1"

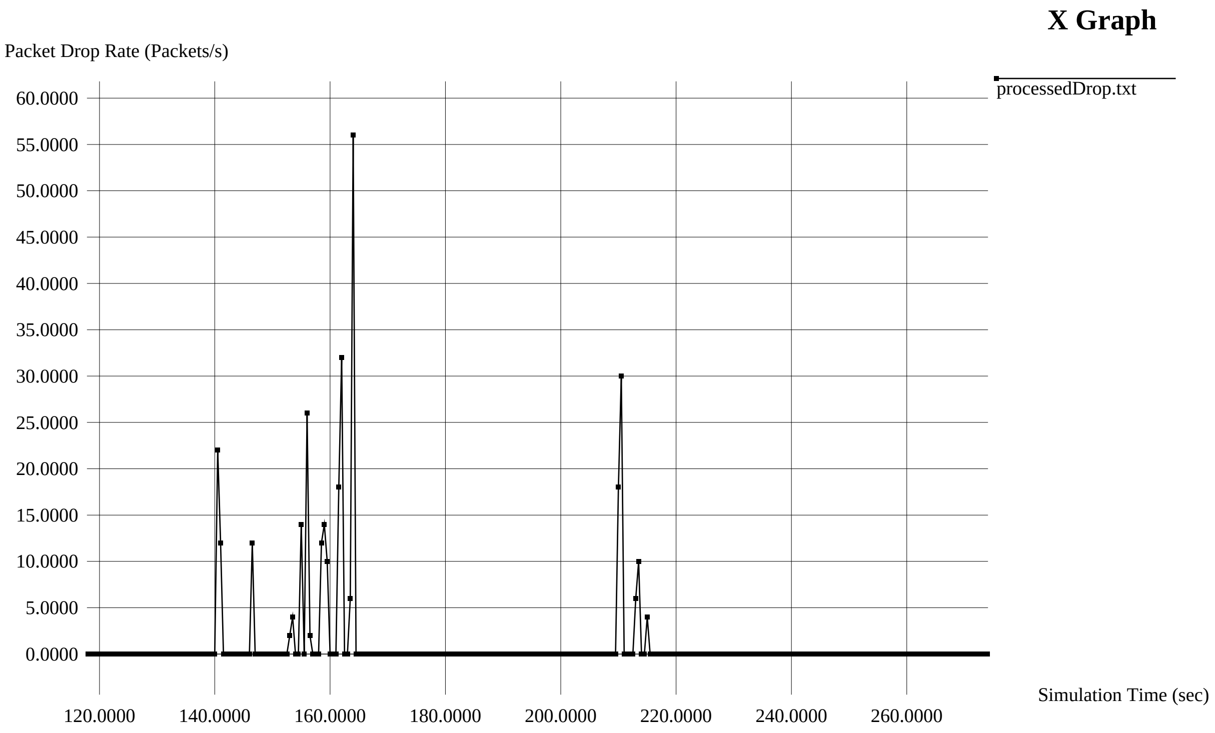


Figure 3.8: Packet Drop Rate for "Six Nodes, Simulation 2"

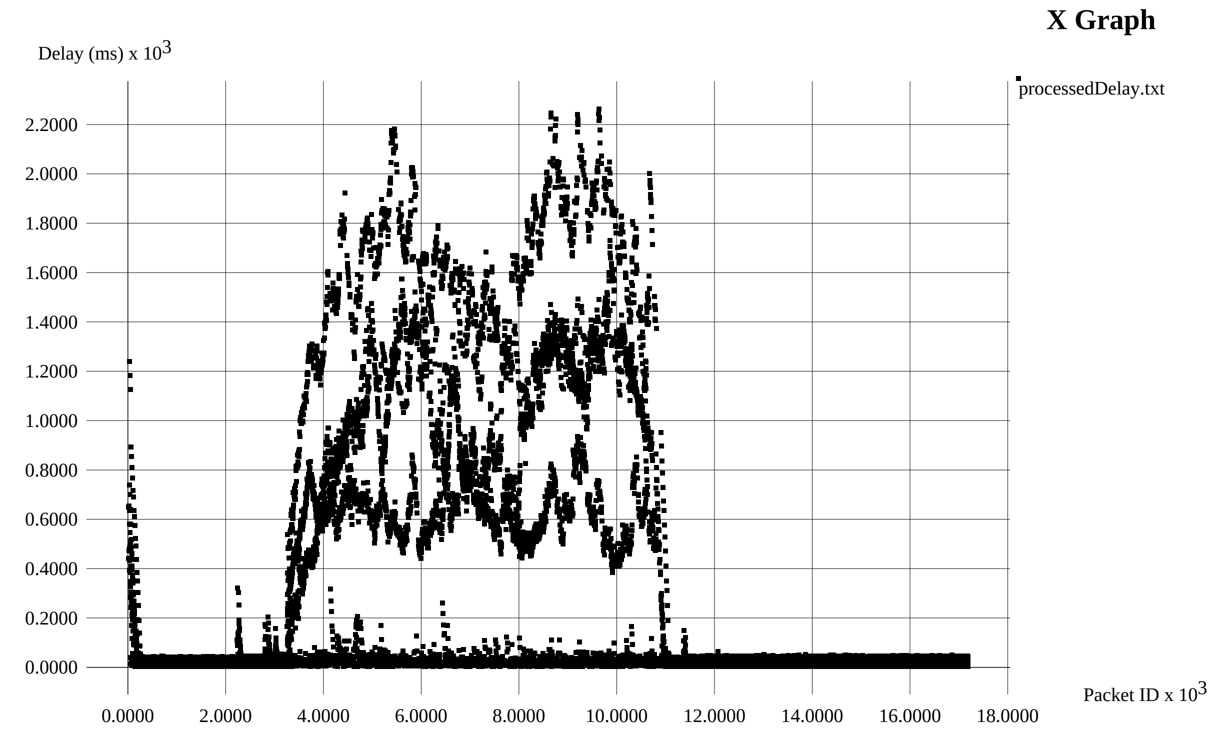


Figure 3.9: Packet Delay for “Six Nodes, Simulation 1”

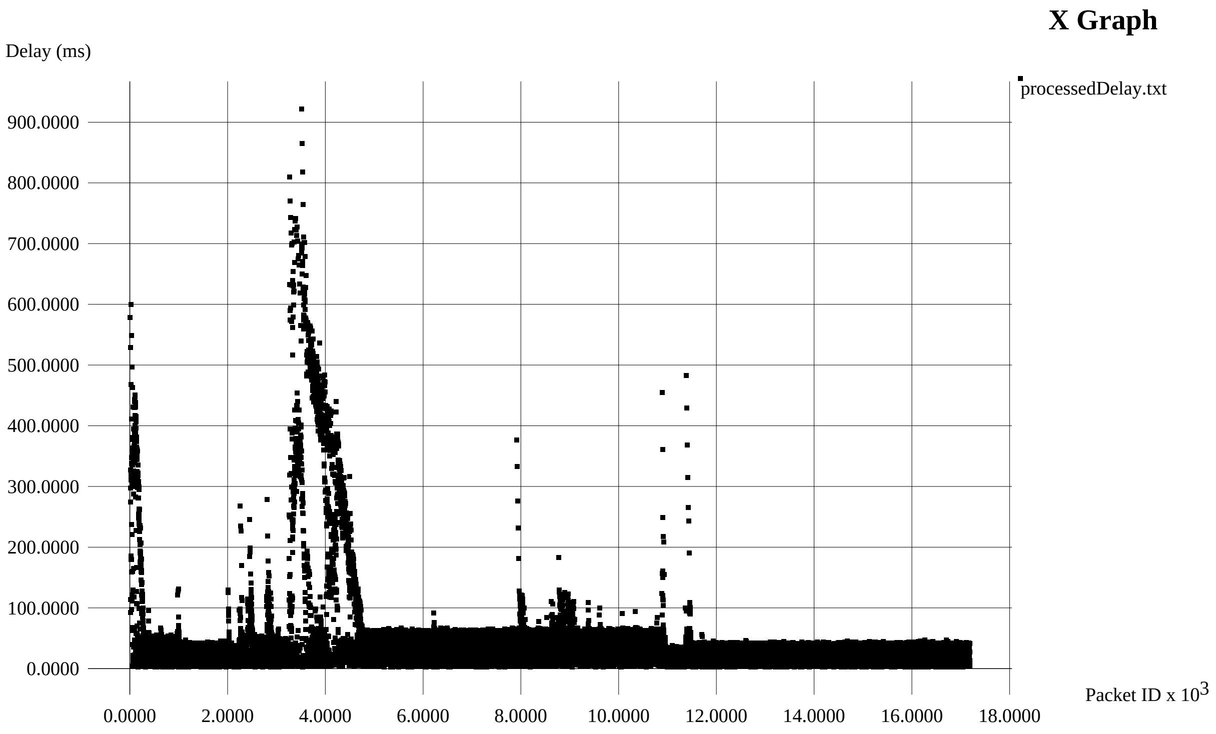


Figure 3.10: Packet Delay for “Six Nodes, Simulation 2”

3.4.5 Two Nodes Apart

The last scenario simulated consisted of two nodes moving in the same direction, but with a considerable distance between them, communicating with each other. The simulation ran for 50 seconds, from 200 seconds to 250 seconds simulation time. The two nodes started approximately 400m apart but were close together by 225 seconds simulation time due to traffic lights. By the end of the simulation, there was approximately 200m between them again. The scenario produced different results based upon the seed of the random number generator. Two sets of simulation results are given in table 3.5. The average delay for both simulation 1 and 2 was fairly low and the number of received packets with a delay greater than 150ms was also fairly similar for both. The main difference between the two simulations was the number of packets dropped. Simulation 1 had a packet drop rate of over 2% while simulation 2's packet drop rate approached 6%. The packet drop rate for simulation 1 and 2 is shown in figures 3.11 and 3.12 respectively. The majority of dropped packets were dropped in the first fifteen seconds of both simulations but the overall number of packets dropped during simulation 2 was much greater than those dropped during simulation 1.

Table 3.5: Simulation Results for "Two Nodes Apart"

	Packets Transmitted	Packets Dropped	Percent Dropped Packets	Packets Received	Packets With Delay Over 150ms	Average Packet Delay
1	1564	35	2.24%	1529	29	42.7ms
2	1564	92	5.89%	1472	33	33.4ms

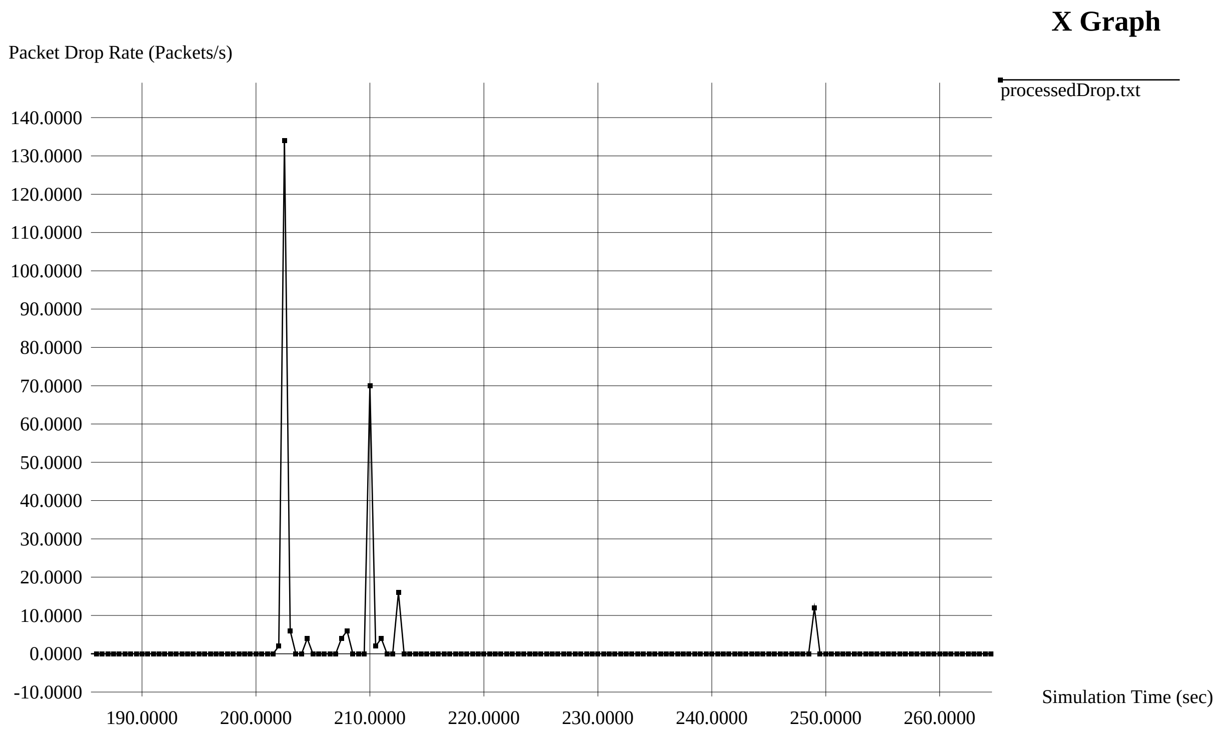


Figure 3.11: Packet Drop Rate for “Two Nodes Apart, Simulation 1”

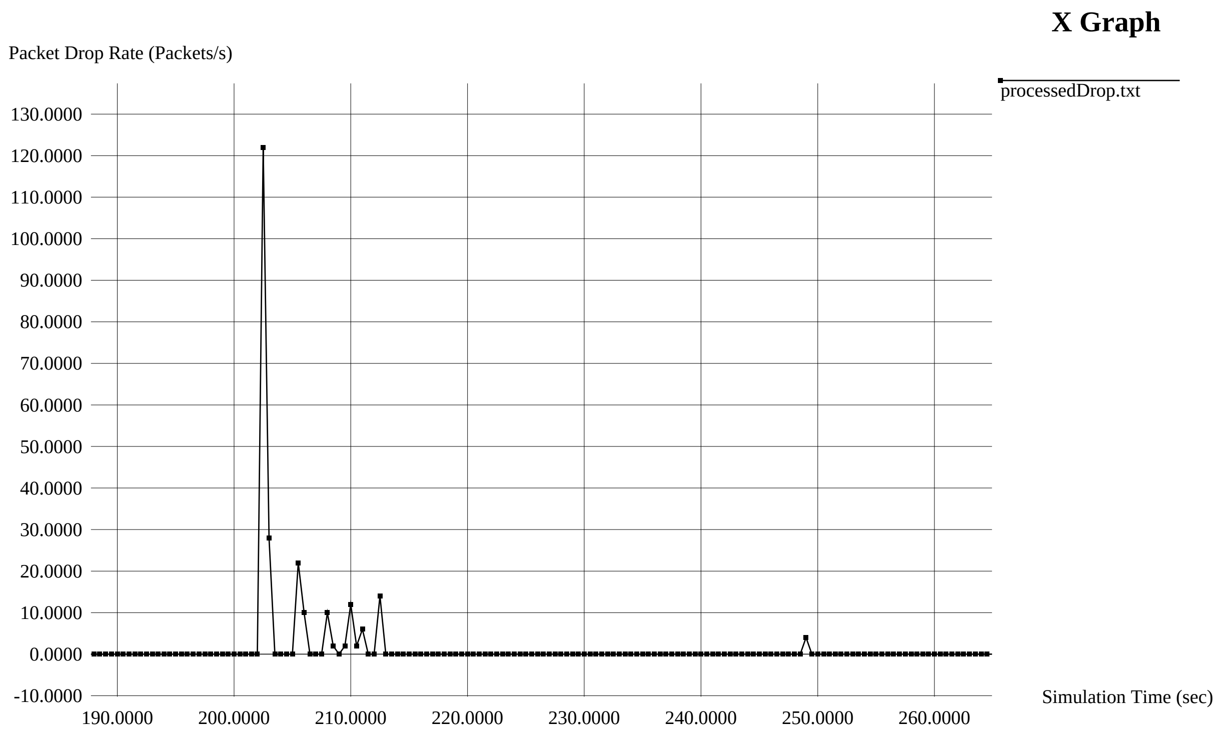


Figure 3.12: Packet Drop Rate for “Two Nodes Apart, Simulation 2”

3.5 Analysis of Results

In the scenario "Two Nodes Close Together", the two communicating nodes were travelling within a one-hop radius of each other for most of the simulation. At the start of the scenario, however, the nodes were a bit further apart. As can be seen from figure 3.1, the majority of packets dropped during this scenario were dropped as the nodes drew closer together and the communication path between the two nodes was being redefined. It can be expected that the main reason for packets dropping in AODV, assuming that a path exists between the communicating nodes exists, would be due to links breaking necessitating new routes to the destination to be discovered.

The scenario "Two Nodes Moving in Opposite Directions" was an extreme scenario which forced the AODV paths to be continually readjusted. By analysing figures 3.2 and 3.3 it can be seen that there was actually two significant periods during the scenario where there appeared to be no communication route between the two communicating nodes. The throughput and packet delay between the two nodes was fairly constant and quite good between 190 seconds and 215 seconds simulation time. This is where the two nodes were passing each other and they were within a one-hop radius of each other. They were actually stopped at the opposite side of the same set of traffic lights for some of that period. The last period of communication between the two nodes is interesting. The throughput and packet delay was fluctuating significantly. At this point, the two nodes were stopped at traffic lights at the opposite ends of the street to each other and were dependant on intermediary mobile nodes to forward packets through multiple hops. The routes between the two nodes were very unstable and AODV would not be able to support the requirements of FPS games under such conditions.

The "Five Nodes" scenario produced some interesting results in that the requirements of a FPS game seemed to be satisfied. By analysing figure 3.4, it can be seen that nearly all packets had a delay of approximately 50ms of less. This is not entirely unexpected. Since most of the nodes moving in the same direction along the street would be travelling at approximately the same speed, the distance between the nodes would be fairly constant meaning the routes set up by AODV can be stable at times.

The "Six Nodes" scenario was similar to the "Five Nodes" scenario except that it produced rather varying results. As can be seen from figures 3.5 and 3.6, while the throughput for simulation 2 was fairly constant throughout, the throughput for simulation 1 had a significant period where overall throughput fell. By comparing figures 3.7 and 3.8, it can be seen that many packets were being dropped during this unstable period in simulation 1 while simulation 2 actually had very few packets dropped during this same period. As can be seen in figure 3.9, the packet delay for simulation 1 during this period was chaotic. However, it is likely that not all client-server streams experienced this degraded performance as can be seen by the many packets still being transmitted with short delays. The packet delay for simulation 2, shown in figure 3.10, actually shows a similar distribution to the "Five Nodes" scenario packet delay graph (figure 3.4) with the exception that the spikes are much more severe in the "Six Nodes" scenario. A likely reason for the significant difference in performance of AODV in the "Six Nodes" scenario for simulation 1 and 2 is that the routes selected for packet transport in simulation 1 were far less stable than the links chosen in simulation 2. This reinforces the point that link stability is a key factor in AODV's suitability to support multi-player games in VANETs.

The "Two Nodes Apart" scenario sought to explore the difference in results for AODV if the distance between the nodes was increased. Despite the increased number of hops that packets had to traverse, the packet delay was still low and very capable of supporting FPS games. There was a fair degree of variation in the rate of dropped packets for this scenario, dependant on the seed of the random number generator. The packet drop rate for simulation 1 and 2 is shown in figures 3.11 and 3.12 respectively. Most of the packets were dropped between 200 and 215 seconds simulation time. Once again, the likely reason for the difference between the two simulations is that the stability of the links chosen to forward packets in simulation 1 were more stable than those chosen in simulation 2.

The primary reason for a significant number of dropped packets or high packet delays in AODV is that the links supporting packet forwarding are unstable and constantly breaking. AODV must maintain entire routes from source nodes to destination nodes in order to forward packets through a network. If the mobility of nodes is continually causing the links which are supporting the packet forwarding process to break, AODV becomes incapable of satisfying the requirements of multi-players games. AODV's ability to support multi-player games in VANETs is, therefore, dependant on the specific mobility pattern of the vehicles involved. It can only support multi-player games when forwarding nodes are sufficiently static with respect to other forwarding nodes in the network, such that the links between these nodes last long enough to keep the packet drop rate and the packet delay sufficiently low. Position-based routing protocols may be better suited to supporting multi-player games in VANETs since they do not maintain the specific routes that packets travel but instead make routing decisions on-the-fly.

Chapter 4

Conclusion

Lots of work has previously been done on developing and testing routing protocols and some of this work specifically investigated the use of routing protocols in vehicular ad hoc networks (VANETs). This dissertation analysed different routing protocols with VANETs in mind and also investigated the network requirements of multi-player games. This allowed the routing protocols to be judged for suitability towards multi-player games running inside VANETs. Simulations were run in order to assess the suitability of a representative example of a topology-based routing protocol. The Ad hoc On-Demand Distance Vector (AODV) routing protocol was chosen for this simulation since it is a very standard routing protocol.

4.1 Suitability of Routing Protocols for Multi-player Games in VANETs

During the simulations performed for AODV, it was found that the stability of the links chosen to forward packets was a governing factor for whether or not AODV could support the requirements of multi-player games. If the forwarding route is being continually broken and reformed, AODV cannot keep the packet delay and the packet drop rate sufficiently small for multi-player games. Due to the mobility of vehicles in VANETs, which can lead to rather dynamic network topologies, routing protocols which require the maintenance of entire routes from source to destination nodes are not particularly suited to supporting multi-player games. As position-based routing protocols do not require the maintenance of such routes, they may be a better choice of routing protocol when it comes to supporting multi-player games in VANETs.

Greedy packet forwarding and contention-based forwarding approaches show advantages over other position-based forwarding approaches, based upon prior research, when it comes to satisfying the requirements of multi-player games. They are efficient in terms of overhead and packets follow a near-optimal path. They provide satisfactory throughput and tolerable packet loss ratios in sufficiently dense graphs. The reactive location service (RLS), set up to use binary search, provides a location service for these forwarding strategies that can tolerate high speed nodes and scales very well. In order to minimise the flooding of query location packets, binary search takes advantage of the fact that, in order to achieve the required latencies for multi-player games, communicating nodes are likely to be close to each other. Multi-player games should be playable in VANETs in the near future when VANETs become mainstream. More research into routing protocols and other factors that affect the playability of multi-player games in VANETs is required before this can happen but it is likely that the network requirements of multi-player games can be satisfied.

4.2 Future Work

There are multiple areas in which research is still required in order to allow multi-player games to be played in VANETs. Several of these are discussed here.

- It is important to simulate, or perhaps implement, position-based routing protocols in VANETs for the sorts of traffic that multi-player games produce. Without this, protocols cannot be properly assessed for their the suitability towards multi-player games.

- It would be interesting to discover what other sorts of traffic might be sharing the network and what percentage of the remaining bandwidth games could actually take. How many simultaneous players or games could then run on the same network?

- Sufficient durations for multi-player games can only be expected in VANETs when the vehicles involved in any one game are moving in a similar direction for a sufficient period of time. Consequently, a selection system is required such that a player only joins a game where a sufficient connection duration, as needed for the game in question, is predicted. Development of such an algorithm is another possible path that future work could take.

Bibliography

[1] James Kuai and Peter Shackelford. *Casual Gaming Market Update*. Parks Associates, 2007.

[2] Mark Claypool, David LaPoint, and Josh Winslow. Network analysis of counter-strike and starcraft. In *IPCCC 2003: Proceedings of the 22nd IEEE International Performance, Computing and Communications Conference*, pages 261–268, 2003.

[3] Tom Beigbeder, Rory Coughlan, Corey Lusher, John Plunkett, Emmanuel Agu, and Mark Claypool. The effects of loss and latency on user performance in unreal tournament 2003®. In *NetGames '04: Proceedings of 3rd ACM SIGCOMM workshop on Network and system support for games*, pages 144–151. ACM, 2004.

[4] Lothar Pantel and Lars C. Wolf. On the impact of delay on real-time multiplayer games. In *NOSSDAV '02: Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pages 23–29. ACM, 2002.

[5] Wu chang Feng, Francis Chang, Wu chi Feng, and Jonathan Walpole. A traffic characterization of popular on-line games. *IEEE/ACM Transactions on Networking*, 13(3):488–500, 2005.

[6] Francis Chang and Wu chang Feng. Modeling player session times of on-line games. In *NetGames '03: Proceedings of the 2nd workshop on Network and system support for games*, pages 23–26. ACM, 2003.

[7] Kuan-Ta Chen, Polly Huang, and Chin-Laung Lei. Game traffic analysis: an mmorpg perspective. *Elsevier's Computer Networks*, 50(16):3002–3023, 2006.

[8] Tobias Fritsch, Hartmut Ritter, and Jochen Schiller. The effect of latency and network limitations on mmorpgs: a field study of everquest2. In *NetGames '05: Proceedings of 4th ACM SIGCOMM workshop on Network and system support for games*, pages 1–9. ACM, 2005.

[9] Paul Bettner and Mark Terrano. 1500 archers on a 28.8: Network programming in age of empires and beyond. In *The 2001 Game Developer Conference Proceedings*, 2001.

[10] Mark Claypool. The effect of latency on user performance in real-time strategy games. *Elsevier's Computer Networks*, 49(1):52–70, 2005.

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

[12] Miguel Garcia de la Fuente and Houda Ladiod. A performance comparison of position-based routing approaches for mobile ad hoc networks. In *VTC Fall 2007: Proceedings of the 66th IEEE Vehicular Technology Conference*, pages 1–5, 2007.

[13] David B. Johnson and David A. Maltz. Dynamic source routing in ad hoc wireless networks. In *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[14] Charles E. Perkins and Elizabeth M. Royer. Ad-hoc on-demand distance vector routing. In *WMCSA 99: Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, 1999.

[15] T. Clausen and P. Jacquet. Optimized link state routing protocol (olsr). *RFC 3626*, IETF Network Working Group, 2003.

[16] Hannes Hartenstein, Bernd Bochow, André Ebner, Matthias Lott, Markus Radimirsch, and Dieter Vollmer. Position-aware ad hoc wireless networks for inter-

vehicle communications: the fleetnet project. In *MobiHoc '01: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking & computing*, pages 259–262. ACM, 2001.

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

[18] Zygmunt J. Haas and Ben Liang. Ad hoc mobility management with uniform quorum systems. *IEEE/ACM Transactions on Networking*, 7(2):228–240, 1999.

[19] Jinyang Li, John Jannotti, Douglas S. J. De Couto, David R. Karger, and Robert Morris. A scalable location service for geographic ad hoc routing. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 120–130. ACM, 2000.

[20] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97. ACM, 1998.

[21] Michael Käsemann, Hannes Hartenstein, Holger Füßler, and Martin Mauve. Analysis of a location service for position-based routing in mobile ad hoc networks. In *Mobile Ad-Hoc Netzwerke, 1. deutscher Workshop über Mobile Ad-Hoc Netzwerke WMAN 2002*, pages 121–133. GI, 2002.

[22] Michael Käsemann, Holger Füßler, Hannes Hartenstein, and Martin Mauve. A reactive location service for mobile ad hoc networks. Technical Report TR-2002-014, Department of Computer Science, University of Mannheim, 2002.

[23] Prosenjit Bose, Pat Morin, Ivan Stojmenović, and Jorge Urrutia. Routing with

guaranteed delivery in ad hoc wireless networks. *Wireless Networks*, 7(6):609–616, 2001.

[24] Brad Karp and H. T. Kung. Gpsr: greedy perimeter stateless routing for wireless networks. In *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*, pages 243–254. ACM, 2000.

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

[26] Holger Füßler, Jörg Widmer, Michael Käsemann, Martin Mauve, and Hannes Hartenstein. Contention-based forwarding for mobile ad-hoc networks. *Elsevier's Ad Hoc Networks*, 1(4):351–369, 2003.

[27] Young-Bae Ko and Nitin H. Vaidya. Location-aided routing (lar) in mobile ad hoc networks. *Wireless Networks*, 6(4):307–321, 2000.

[28] Ljubica Blažević, Silvia Giordano, and Jean-Yves Le Boudec. Self organized terminode routing. *Cluster Computing*, 5(2):205–218, 2002.

[29] Douglas S. J. De Couto and Robert Morris. Location proxies and intermediate node forwarding for practical geographic forwarding. Technical Report MIT-LCS-TR824, MIT Laboratory for Computer Science, 2001.

[30] Michael Moske, Holger Füßler, Hannes Hartenstein, and Walter Franz. Performance measurements of a vehicular ad hoc network. *VTC Fall 2004: Proceedings of the 59th IEEE Vehicular Technology Conference*, 4:2116–2120, 2004.

[31] Marc Torrent-Moreno, Felix Schmidt-Eisenlohr, Holger Füssler, and Hannes Hartenstein. Packet forwarding in vanets, the complete set of results. Technical Report TR-2006-2, Department of Computer Science, University of Karlsruhe, 2006.

[32] Piyao Liu. Rear-seat multiplayer gaming using peer-to-peer car communication networks. Master's thesis, Trinity College, 2007.

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

[34] Ke Liu. The gpsr implemenation for ns-2 version 2.26 or later. `http://www.cs.binghamton.edu/~kliu/research/ns2code/index.html#gpsr`, 2005.

[35] Next generation simulation (ngsim). `http://www.ngsim.fhwa.dot.gov/`.