AZDV: Map Overlay Routing in Vehicular Ad Hoc Networks

Peter Edward Booth

A Thesis submitted for the degree of MSc by Research

Department of Computer Science

The University of York

December 2009

Vehicular Ad Hoc Networks (VANETs) are ad hoc networks formed between vehicles on the road ways. They allow inter-vehicle data communication without the need for a fixed communications infrastructure. Practical uses of VANETs include broadcasting car safety messages and sharing location specific information including traffic information for navigation.

VANETs are formed between vehicles on road ways and as such have predictable road based paths. Road maps can be leveraged to improve routing by viewing the road ways as the routable data paths. This class of routing algorithm is known as map overlay routing.

This thesis explores map overlay routing. It introduces a new map overlay routing protocol developed for this thesis known as AZDV. AZDV attempts to improve upon existing map overlay techniques, it takes a novel approach to utilizing real-time network data to improve VANET routing performance. The performance of AZDV is evaluated against an established VANET routing algorithm.

Contents

1 Introduction

This past decade has born witness to the rapid rise of wireless data communication to the point where it is almost completely ubiquitous. The world is entering an era of unprecedented connectivity. WiFi, bluetooth and mobile cellular communications are standard features in personal computers and mobile phones alike as the divide between these devices increasingly blurs. However, there is still one major area of modern life in which wireless communication has yet to pervade. Wireless communications remains largely untapped as a communication medium between vehicles on the roadways.

In 1999, the US Federal Communication Commission allocated a dedicated spectrum for inter-vehicle communication (Jia, 2008) in preparation for vehicular networking systems. Since then a wide variety of research on ad hoc vehicular networking has taken place. The term Vehicular Ad Hoc Networks (VANETs) is the term most commonly used to refer to the wireless ad hoc networking between On-Board Units (OBU) in road vehicles.

A VANET does not require a fixed roadside infrastructure. It is a distributed system composed of a number of participating nodes which communicate in localised point-to-point communication. In effect, the communication infrastructure of a VANET is comprised of the participants. They transmit the data and even carry it physically when lacking connectivity. There are many practical uses of this form of localised communication:

- Safety (Yin et al., 2004) (Mak et al., 2005)
  - Collision Warnings
  - Emergency Brake Notifications
  - Signalling via Wireless
- Localised Information sharing
  - GPS Road Traffic Information Distribution/Aggregation
  - Local Advertisments e.g. tourist information, roadside services, petrol prices
  - Traffic Announcements e.g. speed limits, road closures

However, a decade on from the allocation of the dedicated spectrum, VANETs still exist purely in small-scale experimental implementations and computer simulations. VANETs exhibit a number of unique properties over existing wireless LANs or mobile ad hoc networks (MANETs):

1. Network partitioning, meaning that the network exists in partitioned sets of connectivity i.e. pockets of traffic. There are no wireless communication paths between these sets.

2. High mobility of routing nodes. The nodes (vehicles) in a vehicular network are constantly moving.

3. Predictable road-based paths. Vehicles in a road network move along a fixed infrastructure of roads making their movement paths predictable.

These properties create new challenges in terms of establishing useful networks but are exploitable in ways that were not previously possible in MANETs. One such interesting area of development is the domain of routing algorithms. Routing algorithms in previous wired and wireless networks were bound to a routing node orientated design. The predictable road based paths of vehicles make it possible to abstract away from low-level routing designs that only consider small-scale car-to-car interactions, towards viewing roads as the information conduits.

This thesis focuses on routing algorithms that leverage the underlying road topology to perform routing. This class of routing algorithm is known as map overlay routing. It requires abstracting away from a node centric routing approach. A simple example would be to imagine routing a packet from a road junction A to a road junction Z, abstracting away from the exact details of which nodes to route the packet across and instead focusing on the path of junctions that lead from A to Z. This is an example of harnessing the predictable road-based paths of VANET nodes.

This thesis will first introduce the field of vehicular networking. Following this, a unique design of VANET routing algorithm known as AZDV will be presented. AZDV was designed and implemented for this thesis. AZDV is a distance vector (Bellman, 1957) routing algorithm that operates on a map overlay and uses real-time traffic data to route messages efficiently. AZDV makes several unique contributions to the field of vehicular networking. It will be evaluated against an existing VANET routing system which lacks the ability to adapt to real-time traffic.

2 Routing

This chapter is an overview of routing techniques, starting with early routing techniques and moving through to position-aware routing and map overlay routing.

2.1 Scalability

Karp and Kung (2000) state two general major factors that contribute to the scalability of a routing algorithm:

1. The rate of change of topology

2. The number of routers in the routing domain. (In the case of mobile ad hoc networks and vehicular ad hoc networks, all nodes are routers.)

These factors are useful when evaluating the characteristics of any routing algorithm.

2.2 Flooding

Flooding is a static (non-adaptive) (Mcquillan et al., 1978) routing protocol (Tanenbaum, 2003, pp355-357). It is static in that it does not maintain an internal state with which to adapt to changes in the network state. This is opposed to a dynamic routing protocol which maintains state.

A routing node in a wired packet-switched network can communicate directly with a subset of the rest of the network via direct wired links. The rest of the network can be reached via packet forwarding (assuming it is unpartitioned). Flooding can be summarised as *forwarding every received packet over every link except the link from which it was received*. Every packet sent in a network using flooding, will traverse every link in the network. Thus, the packet will reach its destination by traversing the entire network.

Krishna et al. (1997) performs a thorough analysis of the complexity of flooding. They evaluate two routing operations to describe the performance of flooding:

1. Adaptation to a topology change

2. Sending and routing a packet to its destination

Flooding is stateless, so it does not adapt to topology changes. Flooding is invariant to the rate

of change of topology (the first of the two scaling factors presented in 2.1).

The same cannot be said for the second of the scaling factors, *sending and routing a packet to its destination*. Where L is the length of the longest path in the set of shortest paths between all possible routing node pairs. The time complexity of flooding is $O(L)$ (Krishna et al., 1997). This is effectively the best possible performance for the time complexity under this metric. However, the communication complexity is stated as $O(E)$ where E is the number of interconnecting edges between all routing nodes in the network. This is effectively the worst-case communication complexity for *sending and routing a packet to its destination*. This is the downfall of a routing system based upon flooding. Every transmitted packet is duplicated across all routing nodes, using valuable communication bandwidth. Each individual link/edge in the network is required to host all transmitted packets within the network and as the number of members of the network increases so will the number of packets and the quantity of duplication.

Flooding is not often used in large scale practical applications because of its inability to scale but it can be useful for ensuring redundancy in smaller scale systems or for use in the lower-levels of more complex systems e.g. flooding within clusters (Liu and Liu, 2008).

However, as mentioned in Tanenbaum (2003, pp355-357), when ignoring the scalability issues of flooding, it theoretically has the lowest packet delivery delay of any routing algorithm because every packet will take every route available across the network, including the route with the lowest delay. It is therefore a useful benchmark in experimentation.

2.3 Flooding in Wireless Ad Hoc Networks

In mobile ad hoc networks (MANETs), all nodes share the same communication medium. This allows for message broadcasting, in which every node in receiving range of a sender is a recipient of its message. Effectively, if every node retransmitted every packet it received, it would be performing flooding.

There are several difficulties with broadcasting a message via flooding which are referenced under the umbrella of the "The Broadcast Storm Problem" (Ni et al., 1999).

2.3.1 Redundancy

A group of nodes (which are all within range of the same sending node) receive a message and all decide to forward it. For each successive retransmission, a large proportion of the nodes in range of the new sender will already possess the message, having received it from a previous forwarding node. These redundant message transmissions use up valuable bandwidth. Ni et al. (1999) formulate a relation between each additional retransmission of a packet and its average effect on the *expected additional coverage* (EAC). The EAC illustrated in figure 2.1 is the proportion of extra coverage provided by a retransmission in comparison to the original transmission radius. The concept of the EAC allows the cost and benefits of a flooding retransmission to be evaluated.

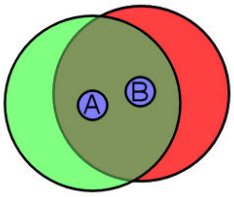


Figure 2.1: The EAC of B over A is illustrated in red.

2.3.2 Contention

In a contention based wireless system such as IEEE 802.11, a group of nodes within wireless transmission range of one another will be competing to use the same wireless bandwidth. In all carrier sense multiple access (CSMA) Tanenbaum (2003, pp255-259) schemes such as this, nodes have to wait for the medium to be free before transmitting which can create problems as the node density gets higher.

2.3.3 Collisions

There will be several instances where multiple nodes attempt to transmit a packet at the same time and their transmissions collide. In IEEE 802.11, after a collision the node back-off time is increased to reduce the probability of a collision on the next send attempt. This will create further burdens on the bandwidth because increased back-off times mean more time used in medium negotiation and less on actual data transmissions.

2.3.4 Improved Wireless Flooding Schemes

There are a number of possible flooding schemes for dealing with the limitations of available wireless bandwidth (Ni et al., 1999), (Williams and Camp, 2002):

1. Probabilistic Scheme - Nodes in a probabilistic flooding scheme retransmit a packet according to a probability shared between all nodes. There is always a risk that no node will forward a particular packet and it will cease to propagate through the network.

2. Counter Based Scheme - Can use the estimated EAC for each successive retransmission to set the counter for optimal flooding retransmissions. When the counter reaches a certain threshold, the node will no longer attempt to retransmit the packet.

3. Distance Based Scheme - Using the distance between the sending and receiving node, and the estimated wireless transmission range, the decision to retransmit can be based upon its potential EAC. An alternative definition by (Williams and Camp, 2002) states that the distance-based scheme use a random assessment delay (RAD) before making the decision to retransmit the packet. After the RAD has expired the locations of the transmitters for each packet are examined, if a transmitter for a particular message is within the given threshold of the current node then the packet is not retransmitted due to the low EAC the

current node has on that particular message.

4. Location Based Scheme - Similar to above except that geometric 2D EAC calculations are performed to assess the transmission coverage gains of a retransmission by a particular node.

Williams and Camp (2002) goes further with a list of neighbour-knowledge based broadcast algorithms:

1. Flooding with self-pruning - Each node maintains a list of all its 1-hop neighbours discovered by "hello" packets. Each broadcast message contains a list of all the neighbours to the sending node. The decision to retransmit can be based upon the difference between the neighbour lists of the sending and receiving node.

2. Scalable Broadcast Algorithm (SBA) - Each node broadcasts all 1-hop neighbours in "hello" packets. Each node maintains a list of all 2-hop neighbours. When receiving a broadcast message, a node will establish a RAD if it finds that it has additional different neighbours to the sending node. At the end of the RAD, if the node still possesses additional neighbours to all the neighbours of the tranmitting nodes, then it will rebroadcast the message.

3. Dominant Pruning - The broadcasting node makes the decision of which nodes will rebroadcast. All nodes have 2-hop neighbour knowledge from a hello packet protocol. The broadcasting node will choose the minimum subset of its 1-hop neighbours which provides coverage for all 2-hop neighbours. Only these chosen 1-hop neighbours are allowed to retransmit the message.

2.4 Dynamic Routing Protocols

(Abolhasan, 2004) Dynamic routing Protocols can generally be divided into two categories:

1. Pro-active - The protocol actively maintains routing information.

2. Reactive - The protocol will construct routes in reaction to a request to send a message.

Distance Vector (DV) routing and Link State (LS) routing are examples of both dynamic and pro-active routing protocols, they were developed for early computer networks (Distance Vector was used in ARPANET (Tanenbaum, 2003, pp357-360)). The aim of dynamic routing algorithms is to forward packets along the optimum route between two nodes with the minimum cost, the minimum cost can be classified in a number of ways including minimum hop count, minimum delay, or the maximum throughput.

2.4.1 Link State (LS) Routing

Each node floods the network with the link-state costs between it and each of its neighbouring nodes. When each node receives a full complement of link-state packets from every node in the network, it can construct a local copy of the entire network graph. A shortest-path algorithm

such as Dijkstra's (Dijkstra, 1959) is then run on this local graph to determine the best next hop for each destination node.

2.4.2 Distance Vector (DV) Routing

Distance vector routing is a dynamic distributed routing algorithm also known as the Bellman-Ford routing algorithm (Bellman, 1957). It was used in the original ARPANET routing protocol (Mcquillan et al., 1978), the Internet RIP routing protocol and Cisco's IGRP (Rutgers, 1991). The algorithm is distributed and asynchronous with the goal of calculating the shortest routing path between all node pairs in the network.

In the basic definition of the algorithm, each routing node maintains a table containing an entry for every routing node known to it in the network. Each table entry maintains the minimum cost path to that destination node, it contains the fields:

1. The destination node Id

2. The next hop node Id

3. The cost to reach the destination node via the chosen next hop e.g. delay, hop count.

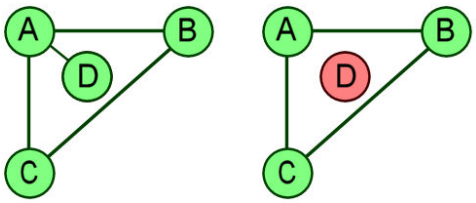
Each routing node within such a DV network will broadcast update packets sharing its internal DV table with all one-hop neighbours. These updates are broadcast both periodically and/or in triggered instances. A node is triggered to broadcast when it receives information which changes its internal DV table and thus triggers it to rebroadcast the changed details (Peterson and Davie, 2000)[pp284-292].

The distance vector algorithm performs according to the *optimality principle* (Tanenbaum, 2003, pp352-353). Briefly stated, given a route from node A to K, and given that the route from A to K can be split into two concatenatable routes from A to B (r1) and B to K (r2). Then if a lower cost route r2 is found, it can be concatenated with the existing route r1 to produce a lower cost route from A to K. From this principle, the distance vector algorithm can find the minimum route to a destination node by finding the minimum cost of (r1 + r2) where A is the routing node, K is the destination node and B is any local next hop node.

The Count To Infinity Problem

Distance vector algorithms can suffer from routing loops including the count-to-infinity problem (Peterson and Davie, 2000), (Tanenbaum, 2003, pp357-360).

Considering a distance vector system of the given topology in figure 2.2(a). Table 2.1 at time 0 shows the state of the routing tables of nodes A,B and C. Consider the situation in which the link from A to D is severed 2.2(b). A will set the distance to D to infinity (effectively unreachable) in its internal routing table as shown at time 1 in table 2.1. Time 2 shows the result of broadcasting this information at the next update informing B and C. Both B and C have internal routing tables which say that D is reachable within 2 hops. When the tables from time 1 are shared between the nodes, they all see a neighbour within 2 hops of D. At time 2 all nodes show a hop count of 3 to node D. This hop count increases successively with each



(a) Node D is connected (b) Node D is disconnected

Figure 2.2: Count-To-Infinity Problem

time 0	A	B	C	D
A	0	1	1	1
B	1	0	1	2
C	1	1	0	2

time 1	A	B	C	D
A	0	1	1	∞
B	1	0	1	2
C	1	1	0	2

time 2	A	B	C	D
A	0	1	1	3
B	1	0	1	3
C	1	1	0	3

time 3	A	B	C	D
A	0	1	1	4
B	1	0	1	4
C	1	1	0	4

time 4	A	B	C	D
A	0	1	1	5
B	1	0	1	5
C	1	1	0	5

Table 2.1: Routing Tables of Nodes A,B and C in the Count-To-Infinity Problem

time step. This is what is known as the count-to-infinity problem. This upward counting will continue until the hop count reaches the networks max hop count (infinity) at which point the node is correctly read as unreachable by all nodes. To sum up, a distance vector system will converge to a link failure in as many steps as the maximum allowed path length in the network system (the representation of infinity). It will converge to any decreased route costs as fast as the information can be propagated.

"the reachability algorithm reacts very quickly to 'good news', and very slowly to 'bad news'" (Mcquillan et al., 1978)

There have been many approaches to partially or fully solving the count-to-infinity problem and other routing loop related problems.

The split horizon (Peterson and Davie, 2000, pp288) solution involves avoiding sending routing information back to a node if it is listed as the next hop node. This avoids sending information back to the originating node of the route and creating a loop. Split horizon with poison reverse (Peterson and Davie, 2000, pp288) is a slightly more pro-active approach using negative reinforcement technique in which a cost of infinity is transmitted to a node regarding any routing table entries of which it is the source. However, these techniques only remedy two-node routing loops and cannot counter a routing loop that happens as a chain of more than two nodes.

Hold down (Mcquillan et al., 1978) is a scheme used to deal with negative routing information. It was introduced in the original routing algorithm of ARPANET. It involves setting a hold down timer on a node when it receives routing information of increased cost. The node will continue to route and update according the original routing data held before the hold down timer was

activated. This is designed to protect the network against any temporary negative fluctuations in routing information.

The Distributed Update Algorithm (DUAL)

DUAL encompasses a family of updated distance vector algorithms which guarantee loop-free status (Aceves, 1993). It uses a method called 'diffusion computation' to uphold the loop free state. The scheme requires that nodes be capable of maintaining one of two states: passive or active. When a router has a stable routing table, it remains in passive mode. However, when a routing node receives an update of an increased routing cost from a neighbour for a particular item that the particular neighbour is the next hop for, then given that there are no other local lower cost routes, the receiving node will enter 'active' mode.

When a node is in 'active' mode it initiates a 'hold down' like state on the affected routing table entry. The state is not the same as 'hold down' because the routing information from the held down state cannot continue to be used for routing as it can in the ARPANET 'hold down'. The node cannot attempt to route a packet to the destination node. The active node then diffuses a shortest path computation to its neighbour nodes by requesting their shortest paths to the destination node of the affected table entry. If any of these nodes are also in active mode then they will also propagate the request outwards to their immediate neighbours (Fall, 2009).

The algorithm, like other DV algorithms still degrades in performance due to negative routing cost changes. It will remain loop-free but its performance can degrade as a result of *hold down* delays. Similar to other DV algorithms, any DUAL based DV algorithms do not perform very well with a rapidly changing topology such as in a vehicular network. The ever changing topology would ensure 'hold down' delays would be common and the high latencies involved with performing updates would significantly reduce the performance of routing (Murthy and Garcia-Luna-Aceves, 1996).

2.4.3 Summary of DV and LS

Distance vector algorithms consume less network resources than link-state algorithms but suffer from slower route convergence attributed to its susceptibility to routing loops (Hong et al., 2002).

Link State protocols consume a larger amount of bandwidth resources than distance vector protocols due to the requirement of periodically flooding the link data of every node across the network. Extra effort is also required in synchronizing the link state packets. However, link state routing does not suffer from the slow convergence problems associated with DV routing.

Both of the previous mentioned dynamic routing methods are unsuitable for routing in wireless ad hoc networks due to the rapid change in topologies and potential large scale of the network which would be engulfed in routing information transfers.

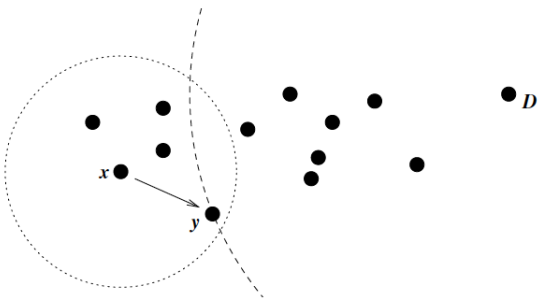


Figure 2.3: Demonstrates a greedy forward from node x to node y with the eventual destination of node D (diagram from Karp and Kung (2000))

2.5 Position Aware Routing

2.5.1 Greedy Perimeter Source Routing (GPSR)

Introduced in Karp and Kung (2000), GPSR is a position-aware routing algorithm. It is one of the most well known position-aware greedy forwarding algorithms. It assumes the existence of a node location lookup service within the network which maps the address of a node to a geographical position. Karp and Kung (2000) proposes a simple beaconing system which allows nodes which are within one hop of one another to exchange position data. Thereby, every node in the GPSR system is aware of the position of all nodes within wireless range. This is with the stipulation that every node in the network has their MAC layer set to 'promiscious mode' which allows all send packets to be received by any node in range.

GPSR is a routing algorithm of two modes:

Greedy Forwarding

This is the primary routing mode of GPSR, it works to successively forward a packet onwards from node to node in a greedy fashion (Figure 2.3). For each forwarding node, the closest node to the destination node in its one-hop neighbourhood is chosen to forward the packet onwards. This allows a packet to progress to its destination covering the most distance whilst minimising the number of hops between nodes taken.

Perimeter Forwarding

Perimeter forwarding is a recovery mode that is triggered in the event that a greedy forwarding candidate does not exist (no neighbour nodes are closer to the destination position than the forwarding node).

This perimeter forwarding mode uses a planar graph representation of the wireless links between the nodes. A planar graph is a graph in which no edges cross one another (Karp and

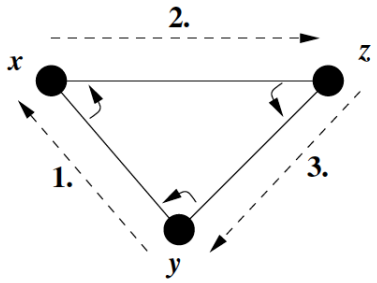


Figure 2.4: The right-hand rule is demonstrated navigating the inside of a triangle by selecting forwarding nodes in an anti-clockwise fashion (diagram from Karp and Kung (2000))

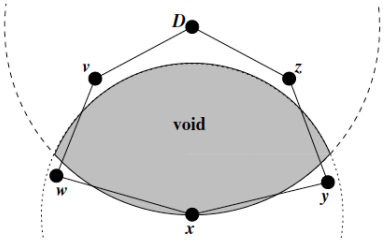


Figure 2.5: The right-hand rule can be used to navigate a packet around the connection void via node x to node w (diagram from Karp and Kung (2000))

Kung, 2000). GPSR can use one of two algorithms to generate these planar graphs from the current known node topology graph, they are Relative Neighborhood Graph (RNG) and Gabriel Graph (GG). The right-hand rule (Figure 2.4) is then used to navigate successive polygonal faces in this graph. Perimeter forwarding is disabled as soon as it produces a graph traversal that produces more candidate nodes for greedy forwarding (Figure 2.5).

GPSR Performance

Within the introductory paper (Karp and Kung, 2000), GPSR is only compared against dynamic source routing (DSR), which is a reactive routing algorithm that utilizes flooding for route discovery and caching for route reuse but which is not position aware. The scalability of GPSR is highlighted with the much reduced routing protocol packet overheads which are of an order of magnitude less than DSR for networks larger than 112 nodes. GPSR also achieves a higher percentage of successful packet deliveries than DSR. The major advantage of GPSR over DSR is its low routing overheads which consist solely of local beaconing.

Lochert et al. (2003) discusses GPSR in the context of vehicular routing and describes several shortcomings of GPSR in this area:

1. Susceptible to routing loops while in perimeter forwarding mode due to node mobility

2. The 'Right Hand Rule' does not find the optimum route when multiple routes to the destination exist, it is biased towards a certain direction.

Tian et al. (2002) describes a simple example in which GPSR will fail to find a route to a destination, even when one exists. Consider a road junction, one exit leading to the destination and other leading away but both routes head in the direction of the destination initially. Then if the wrong route happens to have the better greedy forward, the packet can end up traversing the wrong route. Given the directional bias of the right-hand rule (moves anti-clockwise), there is a chance the packet will never reach its destination. This limitation can only be fixed with a routing algorithm that utilizes knowledge of the road system.

2.5.2 Greedy Perimeter Coordinator Routing (GPCR)

GPCR (Lochert et al., 2005) shares many aspects of the GPSR routing algorithm. Like GPSR, it is largelly stateless, it is composed of a greedy forwarding algorithm and a perimeter forwarding recovery algorithm. It also uses the same beaconing system to allow nodes to identify the position of their 1-hop neighbours. Where they differ is a key bridging point between MANETs and VANETs, which is an extra essential layer of routing abstraction. GPCR demonstrates the concept of overlay routing via the road topology. Whereas GPSR is essentially ignorant of the road system when used in a VANET context. GPCR is designed specifically for VANETs, the assumptions it is based upon rely specifically on the presence of a road network.

GPCR nodes attempt to discern the position of junctions and linear roads in the road network via the positions of other known vehicle nodes in the system. GPCR makes several assumptions:

1. Vehicle nodes on adjacent roads to one another cannot communicate because of obstructions i.e. buildings inbetween.

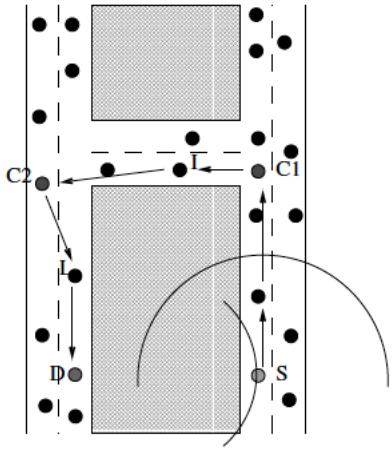


Figure 2.6: Utilizing coordinator nodes, the right-hand rule can be used to navigate around signal obstacles between adjacent roads (diagram from Lochert et al. (2005))

2. Vehicle nodes can communicate with some vehicles on intersecting roads i.e. at junctions.

3. The road system is composed of linear roads i.e. an urban grid pattern

Based upon these assumptions, the vehicle nodes in the system can make deductions about the underlying road network using the known positions of other nodes in the network which were obtained from local beaconing. Each GPCR node calculates the correlation coefficient for each known node against the population of all other known nodes. Node positions with lower correlation coefficients can be said to differ from the shared linear path of the other known local nodes, indicating there is a likelyhood that they are on an intersecting path i.e. at a junction.

The greedy forwarding portion of GPCR uses the concept of coordinator nodes. A coordinator node is a node close to the center of a junction. A node can decide to advertise itself as a coordinator node when it believes itself to be positioned close to a junction. Packets are forwarded to coordinator nodes which being nearest to the centre of junctions, offer the most routing choices for each packet. The coordinator nodes are then responsible for greedily forwarding the message towards its destination. Messages are forwarded greedily along linear roads when no coordinator nodes are identified. The inclusion of coordinator nodes may increase the hop count of GPCR vs GPSR because the coordinator node is not selected in a greedy manner for forwarding so more hops may be needed to cover the same distance as GPSR.

GPCR is shown to outperform GPSR in the metric of packet delivery rate (Lochert et al., 2005). The key difference between the algorithms is the recovery mode used when a forwarded packet encounters a local maximum. To recover from this, both algorithms use the same graph planarization algorithm, the right-hand rule. However, in GPCR, the planar graph representation upon which the right-hand rule is applied is a representation of the road network (Figure 2.6) i.e. junctions for vertices and roads for edges. Whereas in GPSR, the vehicle nodes represent the vertices. This key abstraction allows the recovery algorithm in GPCR to outperform the one used in GPSR for navigating around topology holes.

GPCR still suffers from a lot of the same disadvantages as GPSR. The right-hand rule is biased towards a certain direction and is not guaranteed to find the optimal route when multiple routes exist to the destination. The assumptions upon which GPCR is based also limit its effectiveness beyond simple simulations. Roads are often not linear, the obstruction of communication between adjacent roads cannot be guaranteed and neither can a sufficient sample of nodes needed to calculate junction positions.

GPCR demonstrates the key abstraction of routing via the road network rather than simply considering raw node positions. In particular, it introduces the concept of considering junctions as the key routing points. However, GPCR still uses the Euclidean distance when judging the shortest path to a route-able destination. To make a more accurate calculation it would need actual map data to calculate the possible paths to a destination via the road network. Using map data would also negate the need for statistical techniques to discern the location of junctions, no doubt also leading to better routing performance. This requirement leads to a new class of VANET routing algorithm known as Map Overlay Routing.

2.5.3 Store-And-Forward

When flooding a packet in a partitioned wireless networks such as a VANET, there is no guarantee that it will reach its destination. The network can exist in a partitioned state, such that there is not a multi-hop connection between the sender and destination. With networks such as VANETs, the connected topology changes over time as the nodes move in relation to one another. A strategy is to *store* packets on intermediate forwarding nodes for later replication and retransmission when the network topology alters to possibly produce a path to the destination (Vahdat and Becker, 2000), (Lit, 2005), (Jones and Ward, 2006). This technique can be referred to as store-and-forward, carry-and-forward and epidemic routing. It has been successfully applied to simplistic VANET simulations in Nek (2007). Obviously, without end-to-end connectivity in a network, this technique, unlike flooding, is dependent on both the scaling factors listed in 2.1.

2.6 Map Overlay Routing

2.6.1 Spatial Aware Geographic Forwarding (SAGF)

SAGF (Tian et al., 2002) is another routing algorithm based upon a foundation of 1-hop node discovery beaconing and greedy forwarding. However, SAGF, unlike GPSR and GPCR, utilizes road map data in the process of greedy packet forwarding. In SAGF, each packet carries a road map based route to its destination, this is generated by the source node of the packet. In a process called Geographic Source Route Forwarding (GSRF) (Tian et al., 2002), when a packet is generated, the source node will calculate the shortest route of the packet to its destination using Dijkstra's algorithm on a graph representation of the road map.

To perform these calculations, the road map is stored in the form of graph composed of a set of edges and vertices, where each edge is composed of a pair of vertices. The vertices represent junctions in the road system and the edges represent roads. The cost of each edge is set to the length of the road section between the two composing vertices. From this graph representation, the shortest road paths can be calculated between two vertices in the graph.

Each packet contains an ordered set of vertices describing the shortest path between its source and destination.

$$G = \{V, E\}$$

Each packet in SAGF is forwarded along the shortest road path generated by its source node. Each time a packet is forwarded, the greedy forwarding algorithm is applied along the desired route to the destination. Each time the next vertex in the path is reached, it is removed from the ordered set. Topology holes from the absence of roads are not a problem in SAGF, which gives it an advantage over GPSR. The use of pre-loaded map data in SAGF also gives it the edge over GPCR.

In the event that an SAGF forwarding node cannot locate another node along the pre-generated shortest road path of a packet, SAGF defaults to the greedy forwarding algorithm found in GPSR (without the planar graph recovery method) until it reaches a local optimum, at which point it will attempt to generate another SAGF to the destination node. If both forwarding methods fail, the packet is eventually dropped after a timeout.

SAGF is shown to marginally improve upon the performance of GPSR in the metrics of routing packet overhead, packet delivery ratio and packet delivery delay. SAGF is able to take advantage of the underlying road topology to route packets more effectively.

However, a key limitation of SAGF is it does not handle network fragmentation effectively. It lacks any form of carry-and-forward system, so packets are unable to bridge areas of sparse connectivity and network partitioning. The onboard memory of the vehicular communication unit could be used to temporarily store packets. Vehicles travelling towards the destination of the packet can be used to carry the packet physically. The packet can then be forwarded wirelessly when the carrying vehicle reaches an area of higher connectivity.

2.6.2 Geographic Source Routing (GSR)

Geographic source routing (GSR) (Lochert et al., 2003) is based upon SAGF (Tian et al., 2002). It features the addition of a location service RLS (Käsemann et al., 2002) which is used to discover the position of a node so that SAGF can be used to route to it. This allows node-to-node routing across the network rather than just node-to-location routing.

2.6.3 Vehicle-Assisted Data Delivery (VADD)

VADD (Zhao and Cao, 2006), like GSR and SAGF, is a map overlay routing algorithm with a beaconing system and an underlying greedy forwarding scheme. Rather than take a source routing approach like GSR and SAGF, VADD uses an adaptive probabilistic routing method which allows routing decisions to be taken at each junction rather than at the packet source. The routing decisions are based upon calculated route delays and forwarding node availability. VADD also features carry-and-forward as a routing option in the event that no suitable forwarding nodes are located.

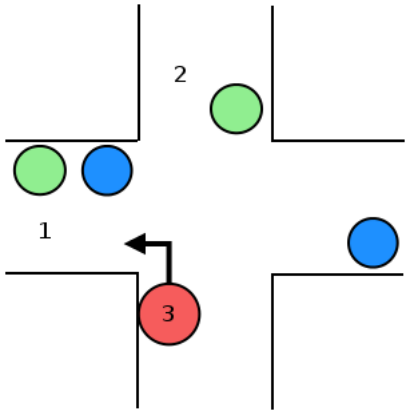


Figure 2.7: The L-VADD routing node identifies the routing priority of the West and North junctions as 1 and 2 respectively. It identifies itself as the 3rd priority routing choice (using carry-and-forward). The greedy forwarding node choices for each path are highlighted in green.

VADD uses delivery delay as the primary routing cost metric. The routing delay of a road section is estimated using the length of the road, the density of traffic along the road, and the average speed of traffic along the road. Therefore, to operate VADD routing, either a real-time traffic measurement system is needed and/or pre-recorded traffic statistics are required to calculate the routing delay of each road. Pre-recorded traffic statistics incur the least overhead but may be inaccurate if actual traffic conditions differ from the average.

VADD is adaptable to VANET network partitioning. GSR and SAGF suffer routing difficulties if the source route of a packet features roads with low connectivity. VADD is not restricted to a single route and re-evaluates the possible routes for each packet at every junction. For a packet at a junction, each route leading away from the junction is assigned a routing priority. The carrying vehicle itself is assigned a priority according to which direction it will be heading out of the junction (carry-and-forward), Diagram 2.7.

VADD has three packet forwarding modes: straight-way, junction and destination. *Straight-way* mode is used when no junctions are in range of the forwarding node, the sole goal of this mode is to greedily forward the packet along the current linear road towards the next junction in the route to the destination. *Junction* mode operates as described in the previous paragraph. *Destination* mode forwarding is enabled when a packet reaches its target destination area. The concept of routing modes in VADD is very similar to the techniques used in GPCR.

VADD has several variants, which differ by their *junction* mode forwarding protocols: Location First Probe (L-VADD), Direction First Probe (D-VADD), Multi-Path Direction First Probe (MD-VADD), and Hybrid Probe (H-VADD). These forwarding probes differ in their criteria for choosing forwarding nodes at a junction.

L-VADD

Location First Probe chooses a node for packet forwarding based upon its location. Starting with the highest priority route for a packet, it searches for nodes along this route, if any are found, then the node furthest along this route, and therefore closest to the destination (greedy algorithm), is picked for forwarding. It repeats this process on the next priority route if the first is empty of nodes. The problem with L-VADD is that it is prone to routing loops. There is nothing to stop a recipient node performing L-VADD and deciding that the node from which it received a packet is the best forwarding node choice. This can happen if the recipient node is moving in the opposite direction to which the packet is traveling i.e. the physical movement of the car undoes any positive ground made by the packet hop.

D-VADD

Direction First Probe avoids the routing cycle problems of L-VADD. It extends L-VADD with a directional clause, which says that to be a forwarding recipient, a node has to be moving towards the destination. This clause stops the packet being routed back to the source node in the junction, avoiding routing loops. D-VADD is typically slower than L-VADD because there is less choice in forwarding nodes due to the directional constraint.

MD-VADD

Multi-Path Direction First Probe is a variation of D-VADD which allows a packet to be sent in more than one direction at a junction. This increases the chance of a successful delivery but increases the network bandwidth overhead.

H-VADD

Hybrid Probe is a combination of L-VADD and D-VADD. H-VADD behaves like L-VADD as default for faster routing. However, when a routing loop is detected, it switches to the loop-free D-VADD for the current junction. This technique was shown to have the overall best performance when taking into account the three metrics by which the algorithms were compared:

1. Delivery Delay
2. Total Data Packets Generated Per Second
3. Delivery Ratio

VADD improves upon SAGF/GSR by introducing the dynamic route generation found in GPCR and GPSR into a map overlay based system. It evaluates potential routes by the calculated packet delivery delay assigned to them. It also introduces the useful mechanism of carry-and-forward.

2.6.4 Landmark Overlays for Urban Vehicular Routing Environments (LOUVRE)

LOUVRE is a map overlay routing algorithm (Lee et al., 2008). Similar to VADD and GPCR, it uses beaconing and performs greedy forwarding between junctions. Where LOUVRE differs is the way it makes routing decisions at junctions. LOUVRE nodes generate traffic density information from the beaconing messages. Each node maintains a local store of node positions from which a beacon message has been received. This local store is used to build density information on local roads, namely the road on which it currently travels. This aggregated density information is then distributed between nodes via a P2P network that exists side-by-side with the VANET network. The density of each edge in the system is distributed within temporal and geographical limits. The shared density information is used by each node to build a link-state table. LOUVRE can be summarized as a map overlay link state (LS) algorithm.

The density information in the LS table is not used to calculate routes in a weighted fashion. LOUVRE essentially uses the LS table to discard edges on the network overlay if they do not meet a certain traffic density threshold. Essentially, the density information is used as a gate on routing paths and the underlying routing costs are still calculated using Dijkstra's algorithm with Euclidean distance as the cost metric, as in SAGF. LOUVRE generally has a higher packet delivery rate than GPSR or GPCR. However, in the metrics of latency and hop count, LOUVRE is outperformed by GPCR although not GPSR (Lee et al., 2008).

LOUVRE does not provide any form of loop detection/prevention. LOUVRE is based upon an assumption that all edge density information in the system will quickly stabilize which means all nodes should have similar LS tables. This makes it less likely that the Dijkstra shortest path routing algorithm will return conflicting results between nodes, which is a cause of loops. Unlike VADD there is no specific route prioritisation algorithm or advanced junction forwarding strategy i.e. D-VADD (Zhao and Cao, 2006) takes into account the direction of the vehicles. Lee et al. (2008) suggest that if carry-and-forward is enabled on LOUVRE, it will simply carry-and-forward along the calculated best paths if no forwarding nodes can be found along the calculated route. LOUVRE uses real-time traffic data in routing calculations. However, it does not use the real-time traffic data to calculate estimated packet delivery delays as done in VADD Zhao and Cao (2006).

3 Requirements

There has been considerable research into routing algorithms suitable for mobile ad hoc networks. However, these algorithms are often ill-suited for use in VANETs. Largely, this is due to the three main properties by which VANETs differ from typical mobile ad hoc networks:

1. Network partitioning
2. High mobility of routing nodes
3. Predictable road-based paths

Most routing algorithms that are designed for ad hoc wireless networks do not attempt to deal specifically with network partitioning or high node mobilities. They will typically face difficulties when encountering such conditions. However, the third unique property of VANETs, the road network, can be exploited to minimize the difficulty caused to routing algorithms by the first two properties. It is the exploitation of known information concerning the underlying road system (such as map data) that makes VANET routing techniques unique.

Section 2.6 contains an overview of the current map overlay routing techniques. SAGF (Tian et al., 2002), VADD (Zhao and Cao, 2006) and LOUVRE (Lee et al., 2008) demonstrate important techniques used in VANET routing algorithms. All of these routing techniques use a weighted graph representation of the road network, in which each edge of the graph represents a road section and the edge weight represents the length of the road section.

SAGF uses Dijkstra's shortest path algorithm (Dijkstra, 1959) to pre-calculate packet routes at their source node. This routes each packet along the shortest Euclidean distance to its destination. LOUVRE and VADD improve upon SAGF by performing dynamic route calculation rather than generating a fixed route from the source.

LOUVRE uses real-time traffic data (traffic density) to further optimize the shortest path routing calculation. LOUVRE does this by eliminating certain road sections from the shortest path calculation, if the traffic density of the road section does not meet the required threshold deemed necessary for forwarding messages.

VADD introduces more advanced junction level dynamic route generation. It uses routing priorities to allow the routing algorithm to adapt to varying actual traffic conditions. Meaning, if there is not a packet forwarding option for the best calculated route, an alternative can easily be picked based upon the calculated route priorities. VADD generates routing priorities based upon estimated packet delivery delay.

VADD leverages both traffic density and average velocity information to estimate the delivery delay along road sections in order to generate and assign routing priorities to all possible routing choices at a given junction (including carry-and-forward). However, VADD does not

provision for the gathering of this density and velocity data which would no doubt add a network bandwidth usage overhead.

From these routing techniques, several goals for a successful routing algorithm can be extracted:

1. Minimize packet delivery delay
2. Improve packet delivery reliability
3. Adapt to real-time traffic conditions and network loads
4. Minimize the bandwidth overhead of the routing algorithm

Of the reviewed techniques, VADD is the only one to use packet delivery delay as the main metric upon which routing decisions are based. Both SAGF and LOUVRE rely primarily upon Euclidean distance as the primary routing metric. Considering the main identified goal is to minimize the packet delivery delay. Using it as the primary routing metric would be the best technique for minimizing it.

VADD estimates the packet delivery delay based upon three parameters:

- Euclidean distance between road section endpoints
- Road section traffic density
- Road section average velocity

However, these parameters are used to estimate network performance without any real measurement of the VANET communication network, only the position and velocity of the network nodes. Relying on the estimated delivery delay could cause problems because similar routing decisions will be made by all routing nodes. This could lead to network congestion along the commonly chosen routes, possibly making alternative routes more viable routing options which would not be reflected by the estimated delivery delay. Improvements could be made to the generation of this metric, in particular, attempting to measure it more directly i.e. by using route discovery packets. This would allow the VANET to adapt to real networking conditions and possibly even balance network loads across multiple routes to the same destination.

The approach taken by the algorithm presented within this thesis is intended to bridge this particular gap. In particular, to measure the performance of the VANET network delays more directly whilst still operating in the same map overlay routing domain as SAGF, LOUVRE and VADD.

4 The AZDV Routing Algorithm

4.1 Introduction

The overall goal of this thesis is to present a new VANET routing algorithm that utilizes the unique properties of VANETs to more effectively route messages. The routing algorithm designed and presented in this thesis is a map overlay routing algorithm. It has been named AZDV in reference to an A-to-Z map. It utilizes real-time routing delay data for route calculation. It is composed of a Distance Vector (Bellman, 1957) algorithm in tandem with an underlying stateless SAGF like map overlay algorithm (Zhao and Cao, 2006), (Lochert et al., 2003). The distance vector featured in the routing algorithm operates in map space rather than node space.

AZDV is a geo-routing algorithm which means that the messages in the system are routed towards a destination location/zone rather than a particular target node. A location lookup/discovery service is required to target a particular node across the road network. The details of how best to implement a location lookup service (if one is even needed) is not covered in this thesis. The design of a location lookup service is a sizable undertaking in itself and is beyond the scope of this thesis. Examples of existing services include GLS (Li et al., 2000), RLS (Käsemann et al., 2002), and PVRP (Rowstron and Pau, 2009).

4.2 Terminology

In this design section there are a number of terms and definitions used to describe the composition and operation of the presented algorithm. To prevent confusion to the reader, the following is a short list containing the definitions of several such terms in the context of this thesis:

- AZDV - The routing algorithm presented in this thesis.
- Node - refers to a vehicle which forms part of a vehicular network.
- Packet - Basic discrete unit of transmitted data in the network.
- Record - An instance of a particular data type which is composed of fixed fields. A packet or data table will contain one or more *records* of the same type.

The rest of this chapter is a description of the technical architecture of the AZDV routing algorithm presented in this thesis.

4.3 Map Overlay Distance Vector Design

Road maps are representable as a graph G, composed of a set of vertices E and a set of edges E that link vertex pairs. Junctions and road segment end-points are represented by the set of vertices V, road sections are represented by the set of edges E (Tian et al., 2002). In a weighted graph, these edges can be weighted to reflect the length of their real-life counterparts.

$$G = \{V, E\}$$

From a weighted graph representation of a road map, it is possible to calculate the shortest geographical paths between map positions using Dijkstra's shortest path algorithm (Dijkstra, 1959).

However, physical distance is not necessarily the best metric to use to generate packet routes through a VANET. Varying traffic conditions and varying network loads can cause equal sized road segments to differ in terms of the data latency and data throughput rates. A cost metric that uses measurements of data latency and data throughput would be more effective than one that uses geographic distance for calculating the shortest path for data across a VANET i.e. SAGF (Tian et al., 2002).

Calculating data transfer delays between a set of fixed positions in a road network is the ideal application of a distance vector algorithm. For an overview of the distance vector algorithm family, see Section 2.4.2. However, Distance vector algorithms have traditionally suffered from major convergence problems in wireless networks that feature high node mobility. The nodes are constantly moving position in relation to one another, sometimes in contact for mere seconds, which makes it difficult to establish any consistent routing paths through the network. This is not the case with AZDV which utilizes static map overlays.

The AZDV algorithm presented in this thesis uses fixed map zones, not routing nodes as the referenced routing entities of the DV algorithm. Typically these geographical map zones would centre around junctions or even road segment end points in the case of longer roads. Static map positions have no mobility thereby avoiding the traditional problem of slow/failed convergence in the presence of node mobility. The network is comprised of high mobility nodes, however, the routing algorithm abstracts away the individual node in favour of addressable zones based upon key features of the road network.

In a traditional node-space DV algorithm, each node acts as its own advocate, each node is a distinct route-able location within the network. In the map space DV algorithm contained in AZDV, the nodes in the network act as advocates for the zones they occupy. It is the zones which are addressable in the map overlay scheme. The road topology is fixed and agreed upon by every node in the network.

Obviously, there are difficulties typically encountered when using DV algorithms, not least when applying them to a new domain such as map space. Problems faced by DV systems in the past include routing loops (including the count-to-infinity problem (Tanenbaum, 2003, pp357-360)) and slow convergence. Before describing the distance vector aspect of my algorithm in more detail, it is important to review DSDV (Perkins and Bhagwat, 1994) in the context of VANETs.

4.3.1 Destination Sequenced Distance Vector (DSDV)

DSDV (Perkins and Bhagwat, 1994) is an update to the Bellman-Ford Distance Vector (Bellman, 1957) routing algorithm. It was designed with ad hoc wireless networks in mind, primarily for dealing with the increased node mobility that is prevalent in such networks. The main goal of DSDV is to reduce the slow convergence time associated with the Bellman-Ford algorithm to allow for stable loop-free routing in an ad hoc wireless setting.

The data records contained within the packets and internal tables of the distance vector algorithm (Bellman, 1957) contain three fields, each record in the table corresponding to a different destination node:

1. Destination node (primary key)
2. Next hop node, in the path to the destination node
3. Routing cost to the destination node

DSDV adds a fourth field: a sequence number which is assigned by the origin node of the record. Each node in a DV network will periodically release a record containing itself as the destination node and a zero routing cost, this is the first step in the transitive route generation that defines DV. In DSDV, for each such new update a node releases concerning itself, it generates a unique sequence number as the fourth field. Obviously for each successive update, the sequence number is incremented.

The purpose of the additional sequence number field in DSDV records is to help prevent the routing loops that are synonymous with distance vector algorithms, the most severe example being the count-to-infinity problem (Tanenbaum, 2003, pp357-360). The added sequence number field is used to prevent older routing information with lower routing costs from being used by nodes for routing instead of newer routing information with more accurate but higher routing costs. A major limitation of the original Bellman-Ford algorithm was that it would react promptly to any improved routing costs but slowly to any worsening routing costs. The addition of the sequence number will allow the networks to react to worsening routing costs as fast as the information can be propagated, rather than as fast as it can loop upwards to the underlying representation of infinity within the network (see section 2.4.2).

In DSDV, the sequence numbers have precedence over the cost metric when making the decision upon whether to replace the information in the internal routing table with routing information from a received DV packet. Any DV packets received with entries containing a newer sequence number than the matching entry currently stored in the internal table, will replace it. Only if the sequence numbers match is the cost metric used to decide the correct routing information to store in the internal routing table, of which the packet with the lowest routing cost is chosen.

The addition of sequence numbers may eliminate the count-to-infinity problem but it introduces some new problems. DSDV can suffer from damping fluctuations (Perkins and Bhagwat, 1994). Damping fluctuations are a regular flip-flopping of routing information every update cycle, it is triggered if routing information of the same sequence arrives worst metric cost order first. This will result in the next hop for the affected destination node fluctuating upon each successive update for the same sequence number, until the best cost routing information for the current sequence number arrives. To counter this problem, DSDV nodes maintain a second table which maintains a record of settling times for each destination node. The *settling time* is the time

between the receipt of the first packet of a sequence corresponding to a particular destination node and the receipt of lowest cost packet of that node sequence corresponding to the same destination node. It is at the end of this interval that the record is stable for the current sequence number.

Each record in the second table of DSDV has three fields. As with the first table, each record corresponds to a different destination node:

1. Destination node (primary key)
2. Last settling time
3. Average settling time

The table functions to provide a release constraint on new routing data. A newly received sequence of routing data must be held for the average recorded threshold time of its destination node before the record is propagated on to other routing nodes.

4.3.2 Map Overlay DV Approach (AZDV)

DSDV (Perkins and Bhagwat, 1994) was reviewed in the previous section because it is the closest analogy in the family of DV algorithms to the map overlay DV algorithm presented in this thesis. Obviously the first major departure of AZDV from any other DV algorithms is that it works in the map space domain rather than the routing node space domain. As with existing DV algorithms, every node within the network will be responsible for maintaining an internal AZDV routing table as well as propagating AZDV routing information packets. Both the routing packets information packets and internal routing table contain the same format of AZDV routing record. An AZDV routing record contains three fields as follows:

1. Destination zone (primary key)
2. Stack of successive next hop zones
3. Origin time stamp of record

The first thing to notice is that the referenced routing primitive (and primary key) in the DV table is a zone rather than a routing node. In the case of this algorithm, a zone is an area of road that is closest to a particular road junction or road segment end point (the centre of which is used as the reference position of the zone). Using road zones rather than routing nodes as the main routing primitive has been used in several recent publications (Lee et al., 2008), (Lee, 2009), (Lochert et al., 2003), (Rowstron and Pau, 2009), (Zhao and Cao, 2006), (Lochert et al., 2005), (Nzouonta et al., 2009), Lochert et al. (2003). Of which, only Nzouonta et al. (2009) utilizes real-time gathered network connectivity data, albeit it is generated in a different manner to this presented AZDV algorithm.

The first two fields of the AZDV routing packet are almost equivalent to the first two fields in Bellman-Ford (Bellman, 1957) DV algorithm records. They represent the destination and the next hop for the packet respectively. In the case of AZDV, the second field is a stack of successive next hop zones that define the first n zone hops in the route to the destination zone rather than a single hop as in the Bellman-Ford algorithm. The max size of the stack can be

altered, the need for a stack is explained in section 4.4.3.

The third field in AZDV is actually a combination of the third and fourth fields used in DSDV (Perkins and Bhagwat, 1994). The third field of AZDV represents the cost of routing to the referenced destination zone in terms of routing delay. However, the third field also acts as a sequence number similar to fourth field of DSDV.

This is possible because of the chosen routing metric used in AZDV: routing delay. The third field of AZDV is the origin time of the AZDV packet from the destination zone to which it refers. The metric cost is therefore the difference between the time the packet was released and when it was received by the current node, the packet delivery delay. The third field is also a sequence number in terms of the release time. The reason only one field rather two is needed is that there is no need to record the received time of the packet which would be used to calculate the actual delivery delay. The first packet received of a particular release time is always the lowest cost due to it being received first and therefore has the lowest delay, negating any need for comparison with future received records of the same sequence number and destination.

As with DSDV, any new packet with a more recent sequence number (release time) has precedence over an older one. A nice side-effect of using this routing delay metric is that damping fluctuations do not occur in AZDV as they do in DSDV because the best cost packet is always received first in AZDV. Therefore, there is no need for a second table to record settling times, or indeed for a settling time mechanism at all.

In the AZDV algorithm implementation, the routing records contained in the internal AZDV table of each node have a fourth field which records the last time a zone entry of the current sequence number was received. This is used in a carry-and-forward mechanism for the AZDV zone records. The purpose is to measure the time since a record corresponding to each routeable zone was received. This allows an AZDV node to make the decision to re-propagate a zone record if it has not been received in a pre-defined amount of time.

In an AZDV system, there are three main types of packet:

- Hello Beaconing Packet - broadcast at regular intervals by each node to advertise their presence and position.
- Data Packet - Service all network requests from the users i.e. carry data from A to B.
- AZDV Routing Packet - These packets are used to generate the AZDV routing information which is then used to efficiently route the data packets.

4.3.3 AZDV Routing Packets

An AZDV routing packet is a collection of AZDV routing records (containing the destination zone, next hop and origin time stamp). There are three main circumstances in which an AZDV routing packet is created and sent:

Periodic

Within the AZDV system, there is a pre-agreed periodic update time which all nodes in the system use. At the start of the update time, all nodes generate an AZDV packet entry. The primary key of the entry is the AZDV zone which the generated vehicle node currently occupies. The hop count of the entry is set to 0 and the origin time is set to the time that the entry was generated. In short, the *periodic* AZDV packet entries are the origin of the routing information within the system. They represent the base case step in generating the routing information.

Forwarding

Upon receiving an AZDV packet containing various entries, a node will retransmit a number of the received entries in a new packet, updating the next hop stack of each record if appropriate. For an AZDV node, the decision of whether to retransmit an AZDV entry after receipt is based upon two conditions:

- As part of the greedy forwarding scheme 2.5.1, a received AZDV packet contains a list of accepted forwarding nodes. If the receiving node is on this list, then it is permitted to forward any records within the received packet.

- Each individual record in the AZDV packet is compared to the equivalent record in the internal AZDV table of the node. If the new entry has a more recent origin time stamp then it replaces the one currently stored in the internal table. If the receiving node is a permitted forwarding node, the updated record is added to the forward queue for retransmission. If the received AZDV entry is of an older or equivalent origin time to the internally stored entry then the entry is not queued for forwarding. The fourth field in the internal table entry (Last Received Time) is updated to the current time in both cases.

Carry-and-Forward

AZDV is designed to run in a fragmented vehicular network. As a result, a carry-and-forward system is needed to allow messages to span areas of sparse connectivity. Otherwise, when a gap in communication occurs a particular AZDV entry would have no options for a forwarding node and would simply stop propagating across the network. This would mean that all routing information would be confined to an area around its origin zone that has an end-to-end wireless connection.

The experimental AZDV implementation in this thesis uses a simple timer based mechanism to allow a packet to continue forwarding after the carrying node crosses an area of disconnection in the network. The fourth field of each entry in the internal AZDV database records the last time an entry concerning each AZDV zone was received.

The mechanism works by periodically checking the time that each entry in the internal database was last received. If the time period between the current time and the last time the entry was received crosses a certain carry-and-forward threshold then the internal entry is set to be forwarded. The constant threshold mechanism is simplistic but functional.

For example, consider a carry-and-forward threshold of ten seconds and a periodic AZDV entry generation period of one second. When a vehicle travels into an area of road with no other vehicles, it will no longer be able to forward any AZDV packets. When it comes back into contact with other nodes it will only forward *periodic* packets and entries received from the new contacts. However, when the records which were received from the other side of the disconnection reach a certain threshold age, the node will begin to re-forward them onwards. This approach allows the AZDV entries to span areas of disconnection. This in turn allows any messages that use AZDV routing to take into account network fragmentation in their routing approach.

4.4 Greedy Forwarding

All low-level routing of packets within the AZDV routing system is handled by a greedy forwarding sub-system. All packets are sent across the road network in a greedy fashion similar to the greedy forwarding method used in GPSR (Karp and Kung, 2000), GPCR (Lochert et al., 2005), SAGF (Tian et al., 2002) and VADD (Zhao and Cao, 2006). There are two greedy forwarding modes in the presented AZDV algorithm:

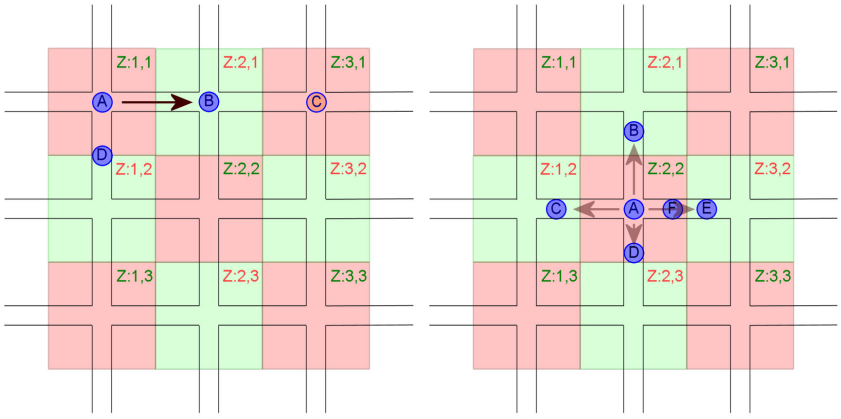
4.4.1 Greedy Unicast Forwarding

4.1(a) This mode is used to forward data packets. It can be used in one of two ways:

- In the absence of any AZDV routing table information with which to route the data packet. This forwarding mode forwards the packets greedily using the same method as SAGF(Tian et al., 2002) i.e. the shortest Euclidean distance to the packet destination.
- If the AZDV table contains a next hop zone for the destination zone of the current data packet, then the data packet is forwarded greedily to that next hop zone. Effectively, AZDV handles the higher-level road based route to the destination and the greedy forwarding algorithm just handles routing the packet between individual zones.

4.4.2 Greedy Flooding

4.1(b) This routing mode is used to propagate AZDV routing packets. In the case of a linear road, the AZDV packet would be greedy forwarded in the two available directions as occurs in the VADD straight road forwarding mode. In the case of a junction, the routing node will attempt to forward the packet greedily to a separate node for each road leading away from the junction, similar to the VADD junction mode (Zhao and Cao, 2006) and PVRP discovery packet flooding (Rowstron and Pau, 2009).



(a) Greedy Unicast Forwarding (b) Greedy Flooding

Figure 4.1: Greedy Forwarding Algorithms

4.4.3 AZDV Lower Level Details

AZDV is quite different in its operation from any other DV algorithm, primarily because it operates in road map space unlike all other DV algorithms which operate in node space. There are a number of low-level details that allow the DV algorithm to operate using this abstraction.

Consider figure 4.2, it shows three vehicle nodes in an idealized grid road system. The routing zones that correspond to their nearest junction are highlighted in green or red and designated with an example coordinate *Z:x,y*. Observe nodes A and C are in zone $(1,1)$ and node B is in zone $(2,1)$.

Now consider figure 4.2 again. When A broadcasts a *periodic* update, assume that B and C are in communications range and that both receive the AZDV packet that contains the AZDV record for the current zone. Node C is in the same zone as A so upon receiving an AZDV packet representing its own zone, it will disregard it. Node B on the other hand, recognizing the packet to be from a zone other than its currently occupied zone, will compare the entry to its internal AZDV table and then add it to the table (assuming its more up-to-date and lower cost than the one currently stored for zone $(1,1)$). As node B received the packet from a node in a differing zone to itself, it will set the next hop of the entry to the zone it was received from. Node B now has a route to zone $(1,1)$ with the next hop set to zone $(1,1)$.

When generating and sending a periodic update packet, the sending node will also commit the update entry to its own internal table. This is useful if the network is sparsely populated and packets have to travel via carry and forward mechanisms. The node will update its internal table when it enters a new zone.

A node needs to modify its internal AZDV table every time it crosses a zone boundary. The next hop stacks of every entry in the internal routing table are modified in this operation. The reason for including a stack of next hop zones rather than just one next hop will be come clear in this section. When a node crosses into a new zone, there are two options for the modification of the stack of next hop zones in each table entry:

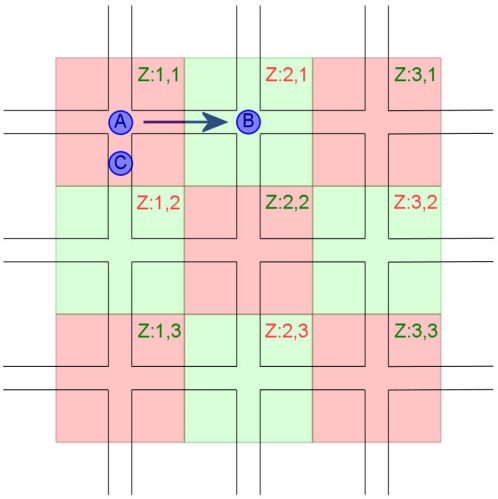


Figure 4.2: Nodes A,B and C in a regular grid based road network.

1. If the node moves into a zone that is currently contained in the next hop stack of any entries, then all zones before and including that zone in the stack are erased, this leaves only zones leading from the current zone left on the stack. If the stack is empty after this operation, then the entire AZDV entry for the destination zone is deleted as it no longer contains next hop information and so is of no further use.

2. If the node moves into a zone that is not in the next hop zone stack, then the zone that was just left by the node is added to the front of the next hop stack. This allows routing node mobility paths to be factored into AZDV routing.

The max size of the stack (before entries at the bottom are erased) is a parameter that is tunable and will be evaluated. It seems the larger the stack size, the better the AZDV algorithm will be able to deal with longer periods of disconnection between nodes (due to sparse traffic and/or a low penetration rate).

4.4.4 AZDV Routing Algorithm Pseudo-Code

Algorithm 1 describes the algorithm in an AZDV node which processes incoming AZDV routing records.

There are several points to note about the algorithm described in Algorithm 1.

- Nodes which are selected to forward a greedy flooding AZDV packet can decide to ignore that request based upon their internal state. In the algorithm, a node will not forward an AZDV packet if it has previously received the same packet in the past. This prevents flooding the same area more than once.

- The call to function *AddHop* essentially pushes the current zone of the receiving node onto the next hop stack of the AZDV record. However, it is a bit more complex in reality

Algorithm 1 The AZDV algorithm for processing a received AZDV record.

```
externalAZDVRecord ← INPUT
forwardFlag ← INPUT {Whether the current node was flagged as a forwarder in the AZDV packet}
internalAZDVRecord ← internalTable.Lookup(externalAZDVRecord.OriginZone())
if internalAZDVRecord = EMPTY then
  internalAZDVRecord ← externalAZDVRecord
  internalAZDVRecord.LastReceivedTime ← Time()
  internalAZDVRecord.AddHop(CurrentZone())
else if internalAZDVRecord.OriginTime < externalAZDVRecord.OriginTime then
  internalAZDVRecord ← externalAZDVRecord
  internalAZDVRecord.LastReceivedTime ← Time()
  internalAZDVRecord.AddHop(CurrentZone())
else if internalAZDVRecord.OriginTime = externalAZDVRecord.OriginTime then
  internalAZDVRecord.LastReceivedTime ← Time()
  forwardFlag ← FALSE
end if
if forwardFlag then
  AddToForwardingQueue(internalAZDVRecord) {Records in the forwarding queue are added to output AZDV packets}
end if
```

because it has to manage possible hop loops and also trim the stack if it goes over the max allowed size.

Algorithm 2 describes the process which occurs on a an AZDV node when it crosses a zone boundary.

Algorithm 2 AZDV algorithm which updates the internal table of a node when the node physically passes into a new zone.

```
for all AZDVRecordRininternalTable do
  if R ≠ EMPTY then
    R.AddHop(CurrentZone())
    if R.LastReceivedTime > TimeOutThreshold then
      R ← EMPTY
    else if R.LastReceivedTime > CarryAndForwardThreshold then
      AddToCarryAndForwardQueue(R)
    end if
  end if
end for
```

Algorithm 3 describes the process of adding zone hops to an AZDV record whilst keeping it in a valid state.

Algorithm 3 AZDVRecord.AddNextHop(Zone): The algorithm which manages adding a new next hop to an AZDV record

```
newZone ← INPUT
if newZone = Record.OriginZone then
  Record ← EMPTY {The Record no longer carries any next hop information because it has
  returned to its origin zone.}
else if newZoneininnextHopStack then
  PopFrontAllafternewZonefromnextHopStack
  if newZone = nextHopStack then
    newZone ← EMPTY {The AZDV record has doubled back on itself so that it no longer
    contains any next hop information}
  end if
else
  PushFrontnewZoneontonextHopStack
end if
while Length(nextHopStack) > MaxStackLength do
  PopBackfromnextHopStack
end while
```

5 Experimentation Introduction

The experimentation was designed to satisfy three main goals:

1. To establish that AZDV is working according to the design parameters.

2. To find and correct any unforeseen issues with AZDV

3. To evaluate the performance of AZDV against an existing VANET routing algorithm, showing the performance advantages of AZDV.

There were two main types of experiment.

Experiment A was designed as a control experiment to verify the performance of AZDV against an existing VANET routing algorithm in a simple scenario. It simulates traffic on a simple linear road. Both AZDV and SAGF were expected to have the same performance under this experiment. It is easier to find and diagnose any unexpected behaviour in a simple scenario such as this.

Experiment B was designed to test the performance of AZDV against an existing VANET routing algorithm under a realistic 2D traffic topology. The overall purpose of the experiment is to highlight the performance advantages of AZDV in comparison to an existing VANET algorithm. It is also designed to reveal any performance issues that may occur with AZDV.

The structure of the experimentation chapters are as follows:

- This chapter introduces the experimental framework, the experiment types, the experiment parameters, and the experimental metrics.

- Chapter 6 presents the experimental results for both experiment A and B. It highlights the areas the AZDV needs improvement.

- Chapter 7 describes the design modifications to AZDV following the results of the first stage of experimentation.

- Chapter 8 presents the experimental results for experiment B following the modifications to AZDV.

5.1 Traffic Traces

A traffic trace is a file containing data on a set of vehicles, recorded over a certain time range to a certain temporal granularity. The data entry corresponding to each vehicle contains at least

the position of that vehicle but may also include other information such as speed. A traffic trace can be derived from real GPS telemetry.

If large numbers of interacting traffic traces are required, a microscopic traffic simulator is required to generate the trace. A microscopic traffic simulator (Krauss) simulates each vehicle as an independent entity, rather than as an aggregated flow as in macroscopic traffic simulators. Simulating the vehicles as individual entities allows complex individual behaviour to be modeled, such as driver reaction time, lane changing, and traffic accidents. There are a number of widely used microscopic traffic simulators in use including MITSIM (Yang, 1996), CORSIM (Halati et al., 1997), SUMO (Krajzewicz et al., 2006b), and TRANSIMS (Smith et al., 1995).

All the traffic traces used to evaluate the routing algorithms within this thesis are generated using the Sumo traffic simulator (Krajzewicz et al., 2003), (Krajzewicz et al., 2006a). Sumo is actively developed by the Centre for Applied Informatics (ZAIK) and the Institute of Transport Research at the German Aerospace Centre (Sum, 2009). Sumo is a microscopic traffic simulator capable of simulating vehicle movement to a granularity of 1 second. It simulates realistic vehicle traces using the *traffic following* model presented in Krauss et al. (1997).

The traffic traces used in this thesis have several key constant features:

1. All simulated roads have three traffic lanes.

2. The speed of the traffic across the three traffic lanes differ by 10 mph each. For example, on a road where traffic is stated to be traveling at 70 mph, only the traffic in the fastest lane is moving at 70mph, the other two are moving at 50 and 60 mph.

In experiment B, both uniform and non-uniform traffic traces were used. The definition of a uniform traffic trace in the context of this thesis is a trace in which the speed and density of traffic parameters were set to be the same across the whole road network. In a non-uniform traffic trace, the traffic speed and traffic density parameters are varied between the road sections.

5.2 Network and Vehicular On-Board Unit Simulation

The network simulator and the simulation of the on-board VANET units are a custom design developed solely for the purposes of this thesis. The two simulators exist as one system, which was developed in C++ for maximum performance.

5.2.1 Network Simulator

The network simulator approximates a single channel CDMA wireless network. It uses a stochastic method to simulate contention for the communication medium. The following properties are enforced by the network simulator:

1. When a node is transmitting, all nodes in wireless range are unable to transmit and in turn will receive all packets transmitted by the transmitting node.

2. When a node is receiving a packet, all nodes in range of the receiver (apart from the

current transmitter) are unable to send but can still receive packets (potentially from a different transmitter to the one transmitting to the blocking receiver).

3. A fixed wireless bandwidth is assumed. When transmitting, the amount of data transmitted is directly proportional to the length of the transmission time. For example, with a bandwidth of 10M bits, to transmit 1 Mega bit of data would require 0.1 seconds. However, a maximum packet size is set within the network simulator to better simulate IEEE 802.11 and stop large continuous transfers.

4. The MAC layer of all nodes in the system are set to promiscious mode (Karp and Kung, 2000) (all nodes can receive all packets sent by any node within wireless range).

Effectively, enforcing the first two criteria mentioned above, creates a simulation of an idealized system in which it is not possible to encounter the *hidden station* problem.

The following assumptions and decisions were made in the design of the network simulator:

1. Each node within the network is simulated as having a fixed wireless radial geometric range. This wireless signal model was chosen for its low computational complexity compared to other models such as shadow-fading models (Bettstetter and Hartmann, 2005).

2. The MAC layer is simulated to the extent that nodes gain access to the transmission medium in a random order. However, there is no simulation of contention featuring network collisions or back-off. If it is deemed necessary, it is a simple modification to apply a basic collision model to the simulator.

In many ways, it is useful to have a simulator which features certain idealized components. It allows new routing algorithms to be evaluated under idealized conditions first before applying them to more complex, closer simulated approximations to a real vehicular network. This also allows difficulties encountered in the simulated routing algorithm to be localized to either high-level or low-level simulation details. There is very little data available on the actual performance of VANET communications so in many cases trying to simulate VANET communications to a high-level of detail could actually harm the relevance of the gathered data if future practical VANETs turn out to vary from the simulated network model used.

5.3 Evaluation Routing Methods

To evaluate AZDV in all performance tests, it was compared to spatial aware geographic forwarding (SAGF) (Tian et al., 2002), (Lochert et al., 2003). See Section 2.6.1 for an overview of SAGF.

SAGF is a map overlay routing algorithm that uses Euclidean distance as the main routing metric. In comparison, AZDV utilizes both Euclidean distance (using the same underlying code as the SAGF implementation in this thesis) and when available, it uses the routes generated in real-time by the underlying AZDV routing system.

Both algorithms share the same underlying local greedy forwarding system based upon the same one-hop hello beaconing protocol (Karp and Kung, 2000). This greedy forwarding system

takes as input a destination position and returns the address of the node best suited for forwarding to that destination (if one exists).

In the case of SAGF, this destination position is the next-hop zone on the generated shortest-path to a packet destination. In AZDV, the next-hop is generated from the data gathered from AZDV route discovery packets. However, if AZDV does not receive any relevant route discovery packets for a particular destination, it defaults to SAGF routing.

The fact that both the algorithms are built upon the same greedy forwarding components allows the experiments to focus on the actual differences between the algorithms. Stated briefly, SAGF is a stateless protocol that is preloaded with the road map. AZDV is not stateless and uses a road map based DV routing scheme to utilize real-time traffic state to route to the destination. AZDV is unicast like SAGF in message transmission but uses a greedy flooding scheme with a carry-and-forward mechanism to distribute the DV routing data. SAGF is a subset of AZDV in terms of functionality.

5.4 Structure of the Experiments

The general structure of each experiment was comprised of five/six(when using AZDV) customizable key components:

1. Road Map
2. Generated Vehicle Trace: Parametrized by
    - Vehicle Density
    - Vehicle Max Speed
    - VANET Penetration Rate
    - Traffic Type
3. Simulated Network System: Parametrized by
    - Max Wireless Channel Capacity
    - Max Wireless Range
4. Routing Method
    - SAGF
    - AZDV
5. Sending module and metric - located on each node, this module was designed to create and send test messages to random positions in the road network. These test messages simulated the network load. The module was also responsible for recording the bandwidth usage of the nodes and making a record of all sent and received messages. These statistics

formed the basis of all experimental data gathered. The module is parametrized by:

- Message send rate

6. AZDV Routing Parameters (if AZDV is the routing method used)
    - AZDV Zone Data Refresh Rate - how often AZDV *periodic* packets are generated
    - Carry-and-Forward Method Delay in AZDV
    - AZDV Max 'Next Hop' Stack Size = 2 (minimum allowed)

These components are tuned to suit the objective of each experiment.

5.5 Stochastic Message Generator

There are three main types of packet which are transmitted in an AZDV system and two types for a SAGF system:

1. User Message (Data) Packets - The goal of the AZDV and SAGF systems is to route these packets to their destination.
2. Hello Packets - Used for neighbour discovery in the greedy forwarding scheme.
3. AZDV Routing Packets - Used for route discovery in the AZDV system. There are three types of AZDV routing packet which are described in 4.3.3.

In the simulations used to compare AZDV and SAGF, the *stochastic message generator* will produce simulated *User Message Packets* for the underlying routing system to handle. It is parametrized by a *Message Send Rate* which is listed as an experimental parameter in 5.4. Each active node in the simulated system contains a *stochastic message generator*, each of which will generate messages at the parametrized rate and assign a random destination zone for the message. Each generated message is then tracked by a metric subsystem as it is forwarded through the network. Vital metrics are recorded for each message to measure the overall performance of the SAGF and AZDV routing systems.

5.6 General Predicted Results

It is a difficult task to investigate the performance of a VANET. It is a distributed system of many interacting components making it difficult to simulate, measure and evaluate. A great many bugs or difficulties in the system were found indirectly from the statistics gathered from experimental runs of the system. The importance of recording the expected results before performing an experiment is invaluable in that regard for determining possible bugs or design errors in the system.

The experimentation evaluates the routing algorithms according to four main metrics:

1. Average bandwidth usage per node
2. Average delivery delay
3. Successful Delivery
4. Average Routing Hops to Destination

5.6.1 Average bandwidth usage per node

AZDV has a routing packet overhead, SAGF has minimal beaconing overheads which are shared with AZDV, this gives AZDV a higher average data usage. Each entry in an AZDV routing packet corresponds to a route to a particular zone/junction in the road system. Each node in the system has to maintain and send one such routing packet for each route-able zone it has received information corresponding to. The AZDV data overhead was therefore expected to be linearly proportional to the number of route-able zones/junctions in the system.

5.6.2 Average Packet Delivery

AZDV generates packet routes according to real-time data gathered by route discovery packets. AZDV was expected to produce faster or equivalent packet delivery rates than SAGF.

5.6.3 Successful Delivery Ratio

This metric records the proportion of successful message deliveries in the VANET system. It was expected that AZDV should be able to outperform SAGF in realistic non-uniform conditions. AZDV was also expected to have some load balancing capabilities, wherein the AZDV routing packets will register larger delays along data congested roads, therefore directing the data traffic along less congested data pathways.

5.6.4 Average Routing Hops

In the context of this metric, a routing hop is defined as a wireless transmission of a user message from one node to another. The hop count is a good measure to determine connectivity in the network. For instance, making a comparison between a sparse and densely populated vehicular network, the dense network will typically have a higher average hop count because there is more connectivity between the nodes and therefore more opportunity to forward a message via wireless communication to its destination rather than use carry-and-forward.

From some preliminary experimentation, it was discovered that this is an effective metric for discovering the presence, frequency and magnitude of routing loops. Large hop counts that are several orders of magnitude larger than the mean/median are clear indicators of a routing loops in which a user message is constantly transfered in a cycle between one or more nodes. High data usage can also provide an indication of routing loops.

5.7 Experiment Types

This section will layout the details and results of each of the planned experiments.

5.7.1 Experiment A: A Simple Benchmark Test Using a Simulated Road Trace Over a Linear Road

This experiment type was designed as a basic verification of AZDV as a VANET routing algorithm. The simulation is conducted upon a simulated linear 5km road. Four separate uniform traffic traces of increasing traffic density have been generated using this road topology.

The main goal of this experiment was to establish a baseline of the routing performance of AZDV. AZDV was not expected to outperform SAGF in any of the metrics within this test. The experiment was designed to highlight any problems with AZDV, which would be apparent if AZDV was not performing within a certain margin of the performance of SAGF.

In this experiment type, there are no junctions on the test road, both AZDV and SAGF are set to split the road into discrete referencable 100m zones. This gives a total of 50 referencable routing zones within the system. This is equivalent to the amount of AZDV routing data for a road system with 50 junctions. Each zone is referenced by its physical central point.

AZDV has been implemented in such a way that it is essentially a superset of the functionality of SAGF. When an AZDV node does not contain any AZDV routing data on the destination available, it defaults to the routing behaviour of SAGF. Therefore, AZDV was expected to mirror the routing choices of SAGF whilst its AZDV routing packets propagate through the network. After the AZDV routing data has propagated out through the network, AZDV was still expected to perform exactly the same routing decisions as SAGF. Due to the linear nature of the test road, there is only one possible route to any destination, so unless AZDV routing data introduces bad information into the system (packets temporarily or permanently travel in the wrong direction), the routing decisions of AZDV were expected to mirror that of SAGF. One function of this experiment is therefore as a basic sanity check of the correct functionality of AZDV.

The performance of both routing methods were expected to be equivalent in terms of the average routing delay times, unless the increased network contention from the AZDV data overhead starts to play a role. The average data transmission rate per node was one area in which AZDV and SAGF were expected to differ. SAGF should use comparatively the least bandwidth because it has no routing overheads. AZDV has data transmission rate overheads due to the distribution of its AZDV routing packets so its bandwidth usage is proportional to the update period of the AZDV system and the number of distinct route-able zones in the road system.

The experiment is designed to be repeated on roads of 2.5km and 10km to test the scalability of AZDV. In particular the scalability of the bandwidth overhead of AZDV can be evaluated.

5.7.2 Experiment B: 2D 1 Kilometer Square Grid Road Trace

This experiment uses a 2 dimensional road map that is 1000m x 1000m in size with junctions vertically and horizontally every 200m to form a grid-based road system. This map is an approximation to a real city (in particular a modern city in the US). While the size might be quite diminutive compared to some US cities (Manhattan is effectively a grid-based city of 25000m x 5000m (Wikipedia, 2009)), it is a good benchmark for the performance of AZDV.

The experiment is performed with two major traffic trace configurations: uniform and non-uniform traffic. Both traffic types were simulated in four levels of average vehicle density. The traffic trace of each linear road section in the grid is simulated separately, which allows the traffic flow on each road to be more precisely controlled. Early experimentation simulating the entire grid as a single traffic system proved difficult with the Sumo traffic simulator. Two variations of this experiment exist, the first features a uniform traffic flow across all roads. The second variation features non-uniform traffic flows in which each linear road section has an random independent traffic density and average speed within the experimental range.

It was expected that SAGF would outperform AZDV on the uniform traffic flow but that AZDV would outperform SAGF for the non-uniform traffic flows. The stateless shortest-path based routing of SAGF essentially performs best in uniform traffic flows. The physical distance based shortest path routing algorithm of SAGF should be accurate for uniform traffic flows where the flow along all equally sized road segments should be equal. However, if the network bandwidth approaches saturation, AZDV may provide a load-balancing role which may result in lower delivery delays and a higher successful delivery ratio than SAGF.

In the case of the non-uniform traffic trace variation, AZDV was expected to outperform the routing capabilities of SAGF. AZDV was expected to be able to identify the best routes using its real-time DV traffic information. Whereas SAGF treats all roads as having uniform traffic. SAGF runs the risk of forwarding packets down roads with low-connectivity.

This experiment is a good test of AZDV routing. It must outperform SAGF in the non-uniform traffic trace experiments in order to prove its effectiveness as a VANET routing algorithm.

6 Experimentation: Stage 1

This chapter documents the first stage of experimentation before the partial redesign of AZDV in chapter 7.

6.1 Experiment A

Experiment A was designed as a control experiment to verify the performance of AZDV against an existing VANET routing algorithm in a simple scenario.

6.1.1 Experimental Parameters

The basic setup of Experiment 1 is outlined in Section 5.7.1. The following is the list experimental parameter settings that were used for this experiment, they follow the structure described in Section 5.4:

- Road Map Length (km)= { 2.5, 5, 10 }
- Traffic Type = {uniform}
- Average Traffic Density (vehicles/km section of road) = { 10.57, 27.45, 60.05, 80.09 }
- Vehicle Max Speed (miles/hour) = 70
- VANET Penetration Rate = 1.0
- Max Wireless Channel Capacity = 1 Mbit
- Max Wireless Range = 50
- Routing Methods = { SAGF, AZDV }
- Message send rate (Hz)= { 0,1,2,4,8,16 }
- AZDV Zone Data Refresh Rate (Hz)= 1
- AZDV Carry-and-Forward Threshold (seconds) = { 10, ∞(equivalent to disabling carry-and-forward) }
- AZDV Max 'Next Hop' Stack Size = 2(minimum allowed)

The main experimental parameters are the routing type, traffic density and message send rate. The generation rate of periodic AZDV routing messages is fixed at 1 Hz. The carry-and-forward threshold of AZDV has two settings. The carry-and-forward setting of infinity is equivalent to disabling the carry-and-forward mechanism. The setting of '10' means that the carry-and-forward mechanism will retransmit any AZDV entry that has not been received in over 10 seconds; Thereby allowing the data to propagate via the internal storage of nodes over network partitions. It is difficult to accurately control the density of a traffic flow within the vehicle simulator, thus the traffic densities used are noticeably irregular in their separation intervals.

The simulation was run for a total of 450 simulated seconds. The first 150 seconds are unmeasured by the metrics, the purpose of the initial time period is to allow the AZDV routing packets to propagate through the system. All user messages created between experiment time 150-250 are measured by the experiment metrics until the experiment ends at experimental time 450. User messages created after time 250 are not measured by the metrics but still contribute to the simulated load on the network while the measured messages move towards their destinations.

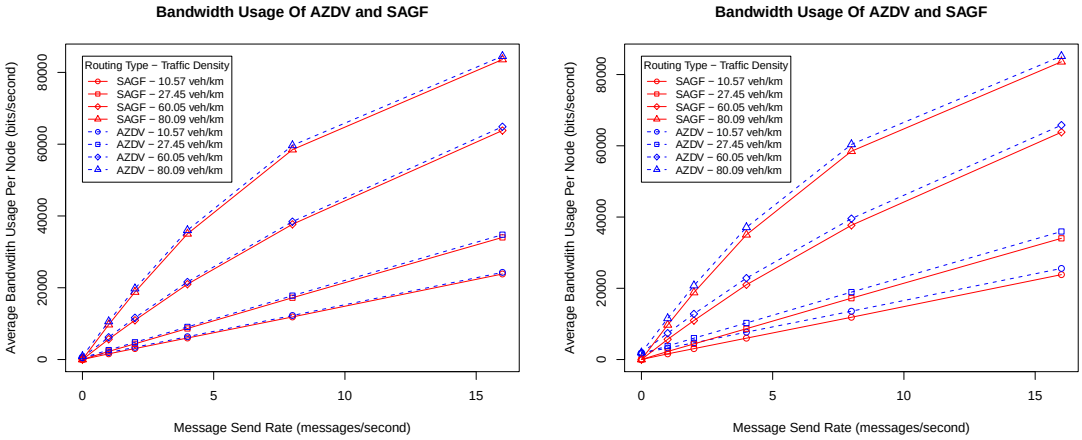
6.1.2 Results

Figure 6.1 shows the bandwidth usage for SAGF and AZDV.

0Hz is the least *message send rate* used to test the system. The *message send rate* refers to the rate of send of simulated uniform size user packets generated by the system. It is worth reinforcing the fact that these simulated user messages are completely separate from the AZDV routing packets. The purpose of the AZDV routing packets are to establish routes which are used to more efficiently deliver the fore mentioned user data packets.

With SAGF, a 0 Hz *message send rate* results in nil bandwidth usage across the system (apart from the Hello beaconing protocol packets) because SAGF is stateless. In comparison, the AZDV system has an AZDV routing packet overhead which it uses for pro-active route discovery so there remains a data overhead, even when user data is not transmitted. The AZDV routing packet data overhead is directly observable when the message send rate is 0. Both graphs in Figure 6.1 show that the AZDV routing packet data overhead is on the order of a few thousand bits or less per node. For higher send rates, the overhead stays constant as predicted. The data overhead is higher for the AZDV method with carry-and-forward than the one without but it is still a constant overhead. This is an indicator that the carry-and-forward mechanism is working, the AZDV routing packets should propagate further under this scheme, therefore more data will be transmitted than the AZDV variant without carry-and-forward. In Figure 6.1, for the higher packet send rates, the bandwidth use of AZDV in the two higher density traces, starts to curve as if leveling off. This is due to the bandwidth cap of 1Mbit, otherwise, the line representing the bandwidth usage would be linear.

Overall, the recorded metric data for SAGF and the AZDV variants differ only by small fluctuations. The hops counts are similar across all methods and do not show any symptoms of routing loops. The message delivery success rates and delivery times of AZDV and SAGF are also very similar and indiscernible at a glance. The fluctuations in results happen mostly in the favour of SAGF, it performs marginally better in all metrics by a few percent. The reasons for this may be due to the AZDV routing packet data overhead which could impact performance slightly.



(a) Carry-and-Forward Threshold = ∞ seconds

(b) Carry-and-Forward Threshold = 10 seconds

Figure 6.1: Exp1 Bandwidth Usage

Figure 6.2 displays the measured delivery delays of SAGF in a scatter plot with a line of best fit. The graph shows that for the uniform traffic trace of this experiment, the message delivery delay is linearly proportional to the distance between the source and destination of the message. This is an interesting result because it confirms that SAGF routing makes accurate routing decisions in uniform traffic. SAGF used Dijkstra's shortest path algorithm (Dijkstra, 1959) with a cost metric of distance, so all generated routing paths are actually the shortest distance between the source and destination. As distance sent and message transit time are linearly related, the SAGF route generation calculates the fastest average route in uniform traffic.

The two simulations with the lowest traffic densities differ in density with each other by around 250% while their message delivery delays differ by less than 10%. Whereas the small increase in density of around 30% between the two simulations of highest density almost halves the delivery delay. The increased connectivity caused by the density increase is the cause. This is true for both SAGF and AZDV. The LOUVRE (Lee et al., 2008) routing algorithm uses traffic density as a secondary routing metric to disable low-density routing paths. This result offers evidence that there is a density range threshold which sees a sharp decrease in delivery delay.

The traffic in the simulation moves at a maximum speed of 70 mph (31.11 m/s). This means a simulated vehicle is able to traverse the entire 5000 meter linear road in 160.7 seconds. This figure matches the delivery delay for the two simulations with the lowest traffic densities. It seems the connectivity is too sparse to forward messages in these simulations, so the messages are *carried* for the majority of their journey.

The standard deviation of the delivery delay in the two lower traffic density simulations is noticeably smaller than in the two higher density simulations. The increased standard deviation can be attributed to the increased use of wireless data forwarding over *carrying*. This is shown in the graph of hop counts, which displays the trend of increased average wireless transfer hops per message in increasing traffic densities 6.4.

It is worth noting that during the early implementation phase, there were a number of bugs in

the routing logic. These bugs caused observable routing loops in the metrics. The *hop count* metric in particular would contain values several orders of magnitude higher than the current values. Another sign of routing loops is a large increase in the bandwidth usage.

Figure 6.3 displays the message delivery success rates. On the whole, the message delivery rate is high. There is a small but noticeable drop towards the end of each of the graphs, which represent the messages sent over the longest possible distance in the system. These drops in delivery success rate could be attributed to message carrier nodes leaving the system before either being able to forward a message or register a received message as successfully delivered. There is no duplication of user messages across nodes so if a vehicle leaves the simulation, all messages contained on it are deleted. In order to send a message to a target 5000 meters away, a node will need to be on the very edge of the simulated linear road, there is a higher probability then that it could go out of bounds before safely forwarding a message. This would explain why the message delivery failure rate is more pronounced for longer distanced sends. It is also possible that some messages have such an abnormally long transit time that they do not arrive at their destination before the end of the simulation.

Figure 6.4 displays the hop counts for all message sent within the system. A hop is recorded each time a message is forwarded via wireless communication. There is a noticeable trend which shows that the hop count increases with the traffic density. Obviously, the higher traffic densities enable more wireless forwarding of messages, which supports the conclusions reached when examining the delivery delays.

This experiment was repeated with the same parameters for two other linear roads of 2.5km and 10km in length. Figure 6.5 displays the average data overhead for AZDV over all three road lengths, with and without carry-and-forward enabled. Without carry-and-forward enabled, the data overhead stays fairly constant due to the fact that the AZDV packets can only be forwarded as far as the immediate node connectivity will permit. With carry-and-forward enabled, the AZDV bandwidth scales linearly with the length of the road and the number of zones. The carry-and-forward mechanism allows the AZDV packets to be forwarded further by utilizing node mobility. On a linear road, with carry-and-forward enabled, the bandwidth overhead of AZDV scales linearly with the number of route-able zones in the system.

6.1.3 Evaluation

Overall, this experiment verifies that AZDV performs as expected under a simulation of idealized uniform traffic along a linear road.

6.2 Experiment B

Experiment B was designed to test the performance of AZDV against an existing VANET routing algorithm under realistic simulated traffic. This is the first run of experiment B. It is performed again in chapter 8 after the partial redesign of AZDV in chapter 7.

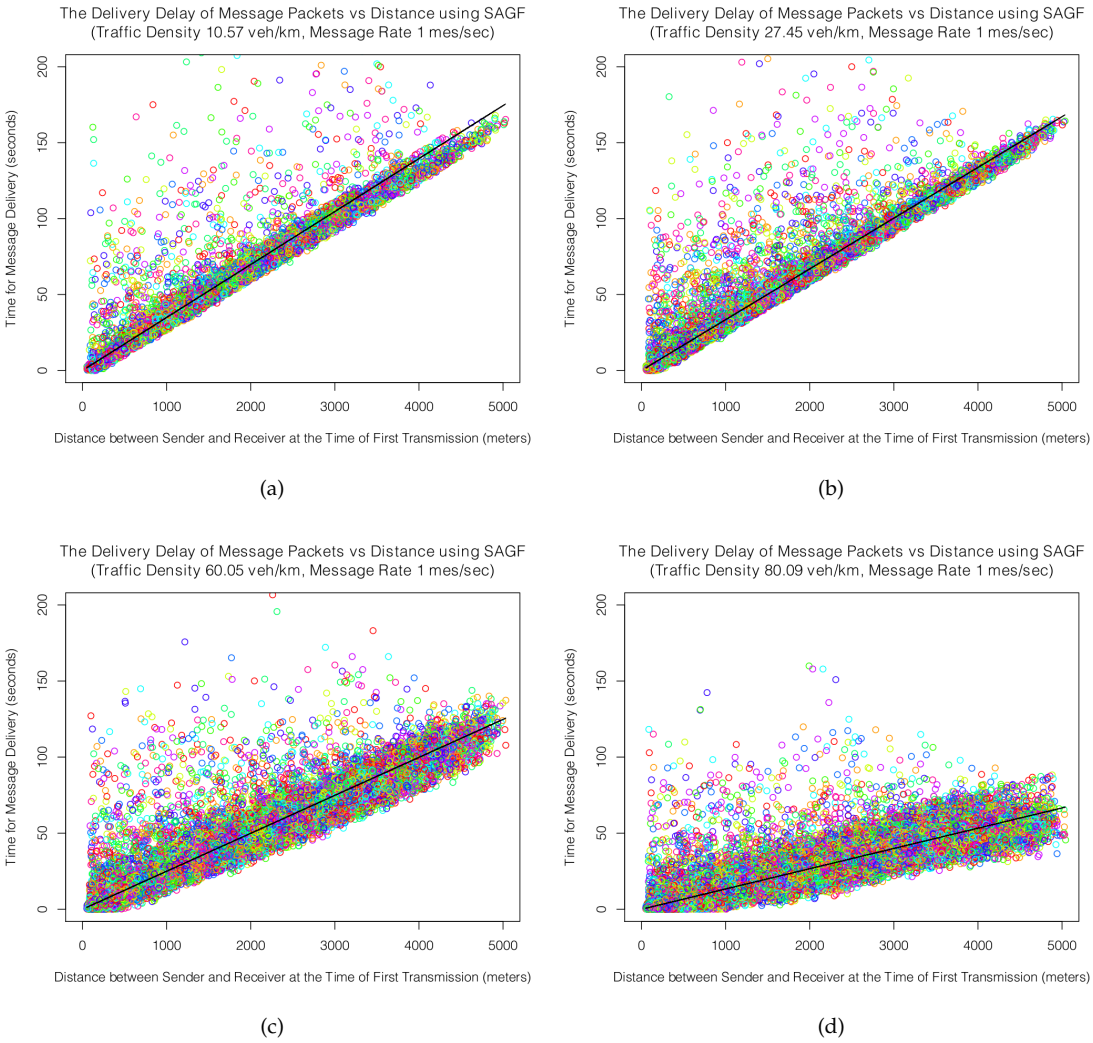


Figure 6.2: Exp1 Message Delivery Delay using SAGF

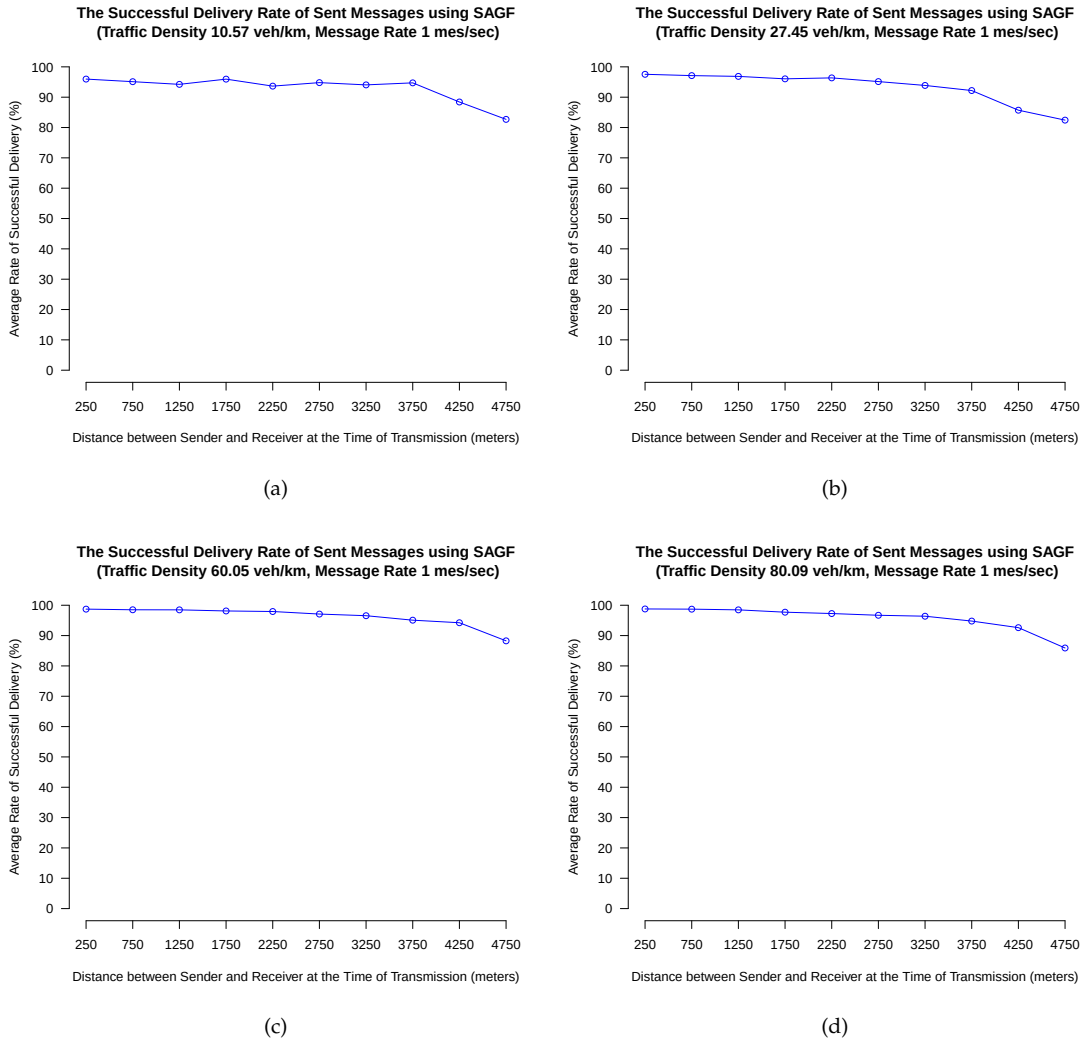


Figure 6.3: Exp1 Message Delivery Success rate using SAGF

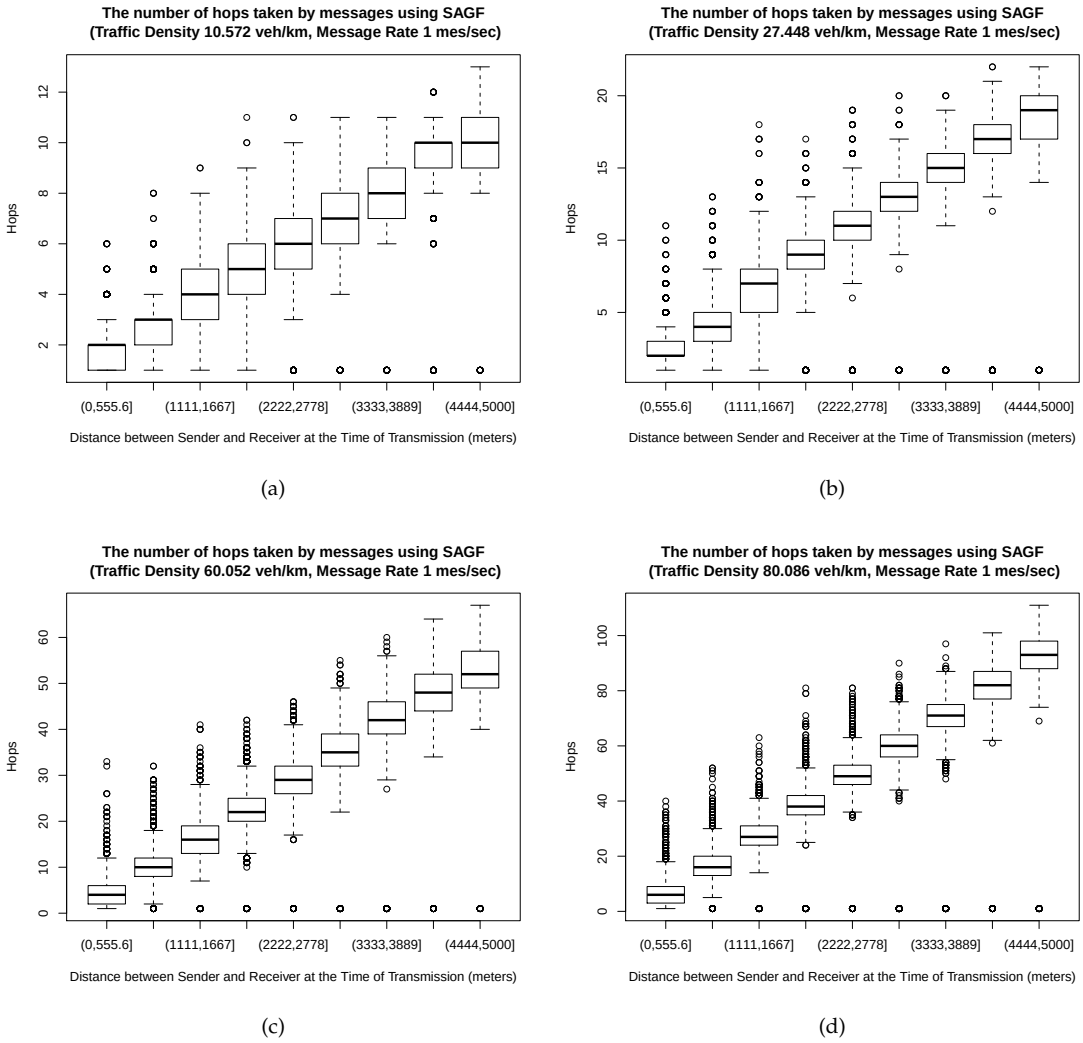


Figure 6.4: Exp1 Message hops per message to get to the destination zone using SAGF

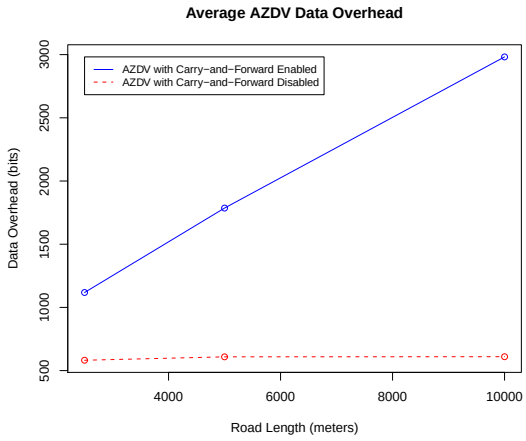


Figure 6.5: AZDV Data Overhead for a Linear Road

6.2.1 Experimental Parameters

The basic setup of Experiment 2 is outlined in 5.7.2. The following is the list of experiment parameters used, which follow the structure in 5.4:

- Road Map Square Grid Dimensions (km^2) = 1
- Road Map Number of vertical and horizontal roads = 6 (The grid is composed of road segments of 200m in length)
- Traffic Type = {uniform}
- Average Traffic Density (vehicles/km section of road) = { 15.93, 35.18, 72.01, 96.37}
- Vehicle Max Speed (miles/hour) = 70
- VANET Penetration Rate = 1.0
- Max Wireless Channel Capacity = 3 Mbit
- Max Wireless Range = 50
- Routing Methods = { SAGF, AZDV }
- Message send rate (Hz)= { 0,1,2,4,8,16 }
- AZDV Zone Data Refresh Rate (Hz)= 1
- AZDV Carry-and-Forward Threshold (seconds) = { 10, ∞(equivalent to disabling carry-and-forward) }

- AZDV Max 'Next Hop' Stack Size = { 2(minimum allowed), 50 }

The simulation was run for a total of 400 simulated seconds. The first 100 seconds are unmeasured by the metrics, the purpose of the initial time period is to allow the AZDV routing packets to propagate through the system. All user messages created between experiment time 100-150 are measured by the experiment metrics until the experiment ends at experimental time 400. User messages created after time 150 are not measured by the metrics but still contribute to the simulated load on the network while the measured messages move towards their destinations.

6.2.2 Results

To produce the following results, the 'Next hop' stack size of the AZDV records was set to 2. The size of the 'Next Hop' stack determines how much of an AZDV record zone traversal path is recorded. In this case, the two previous zones an AZDV path has traversed are recorded. Only the immediate previous zone is used to calculate the next hop of a user message. However, the longer the path, the more likely a routing loop can be detected, which is performed by detecting duplicate zones in the path. The bandwidth metric displayed in Figure 6.6 reveals an alarmingly high data overhead for AZDV which seems to max out the 3 Mbit wireless bandwidth. It displays the symptoms of routing loops in the system.

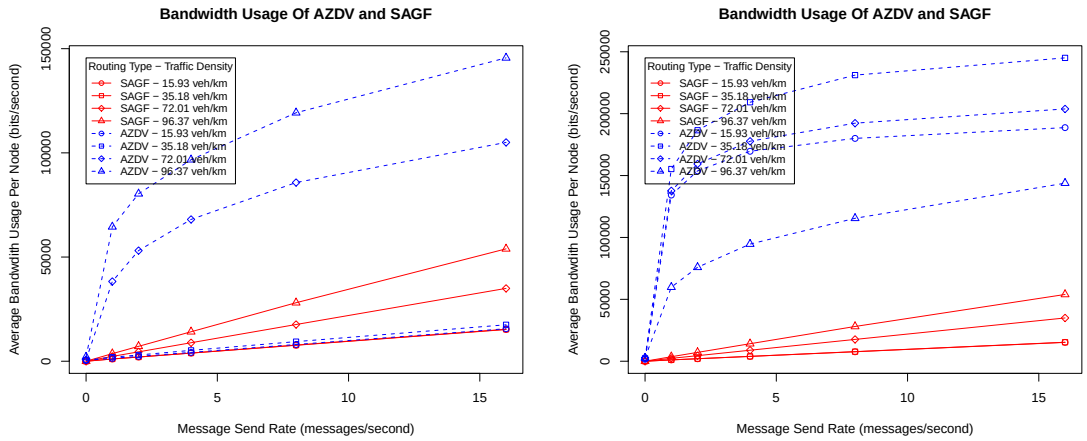
The high AZDV bandwidth overhead occurs in all four traffic densities for the carry-and-forward enabled system. For the AZDV system with carry-and-forward disabled, the large overhead does not occur in the two lower densities. In *experiment 1*, very little message forwarding occurred in similar traffic densities due to the limited connectivity associated with such sparse traffic, in that case the system defaulted to the routing behaviour of SAGF. This is a possible explanation for the low overhead, in the low traffic density runs.

Figure 6.6(a) shows the message hop counts of the AZDV system with carry-and-forward disabled, sure enough, the higher two traffic density simulations have hop counts, several magnitudes higher than expected, indicating routing loops.

The most obvious remedy to any routing loops in the AZDV system would be to boost the maximum limit on the number of next hop zones that are capable of being stored in each AZDV record. It is plausible that the AZDV records could be forwarded in an endless undetectable cycle because their next hop stacks are not long enough to store the cycle of zones. The larger the maximum next hop stack size, the easier to detect and eliminate cycles by identifying duplicate zone entries in the stack. A cycle can be eliminated by popping all items off the top of the stack after the first instance of the duplicate zone. This cycle check is performed each time a new zone is added to the next hop stack.

The maximum next hop stack size is set to 50 for all AZDV records and the experiment is rerun. Figure 6.7 shows very little change, the high bandwidth usage persists. This shows that the routing loops are not primarily caused by insufficient next hop data in the AZDV records.

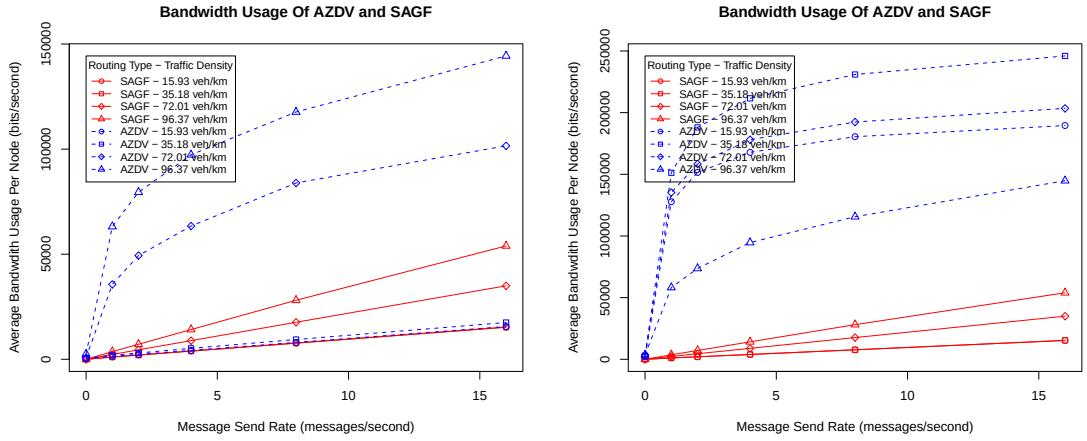
Figure 6.8 and Figure 6.9 both contain four scatter plots which display the hop counts for all the successfully delivered user messages, one plot for each simulated traffic density. The plots are generated from an AZDV system with the maximum AZDV record next hop size set to 50. The only trace not to produce loops is the lowest density run, which registers very few



(a) Carry-and-Forward Threshold = ∞ (CaF disabled) seconds

(b) Carry-and-Forward Threshold = 10 seconds

Figure 6.6: Exp2 Bandwidth Usage with a maximum AZDV Record Next Hop Stack Size of 2



(a) Carry-and-Forward Threshold = ∞ (CaF disabled)

(b) Carry-and-Forward Threshold = 10 seconds seconds

Figure 6.7: Exp2 Bandwidth Usage with a maximum AZDV Record Next Hop Stack Size of 50

hops per message, indicating very low connectivity as expected. As the density increases across the traces, the median hop count seems to become linear with respect to the distance to the destination of the user message. The occurrence of routing loops are visible as outliers with hop counts in the thousands, hence the use of a logarithmic scale. In comparison, Figure 6.10 displays the associated SAGF hop counts, they were obtained under the same experimental parameters and are loop free. This experiment performed with AZDV carry-and-forward enabled raised the number of routing loops further as shown in Figure 6.9.

The metric of message delivery success rate also shows that AZDV is far outperformed by SAGF, as shown in Figure 6.12 and 6.11. SAGF has near a 100% message delivery success rate for the two higher traffic densities.

6.3 Evaluation

Firstly, SAGF is shown to perform well under uniform traffic conditions. It is clear at this stage that AZDV vastly under-performs in comparison to SAGF. Whereas, it was expected that AZDV would only lag slightly behind the performance of SAGF under idealized uniform traffic. The presence of routing loops completely undermines the performance of AZDV. The routing loops cannot be avoided by anything as simple as increasing the next-hop stack size. This suggests the routing loops are not at the map overlay level i.e. are not loops inbetween routing zones. However, the loops do not occur in SAGF so neither do they occur in the underlying greedy forwarding system shared between AZDV and SAGF. The best solution at this stage is to partially redesign AZDV with careful investigation into the cause of the routing loops.

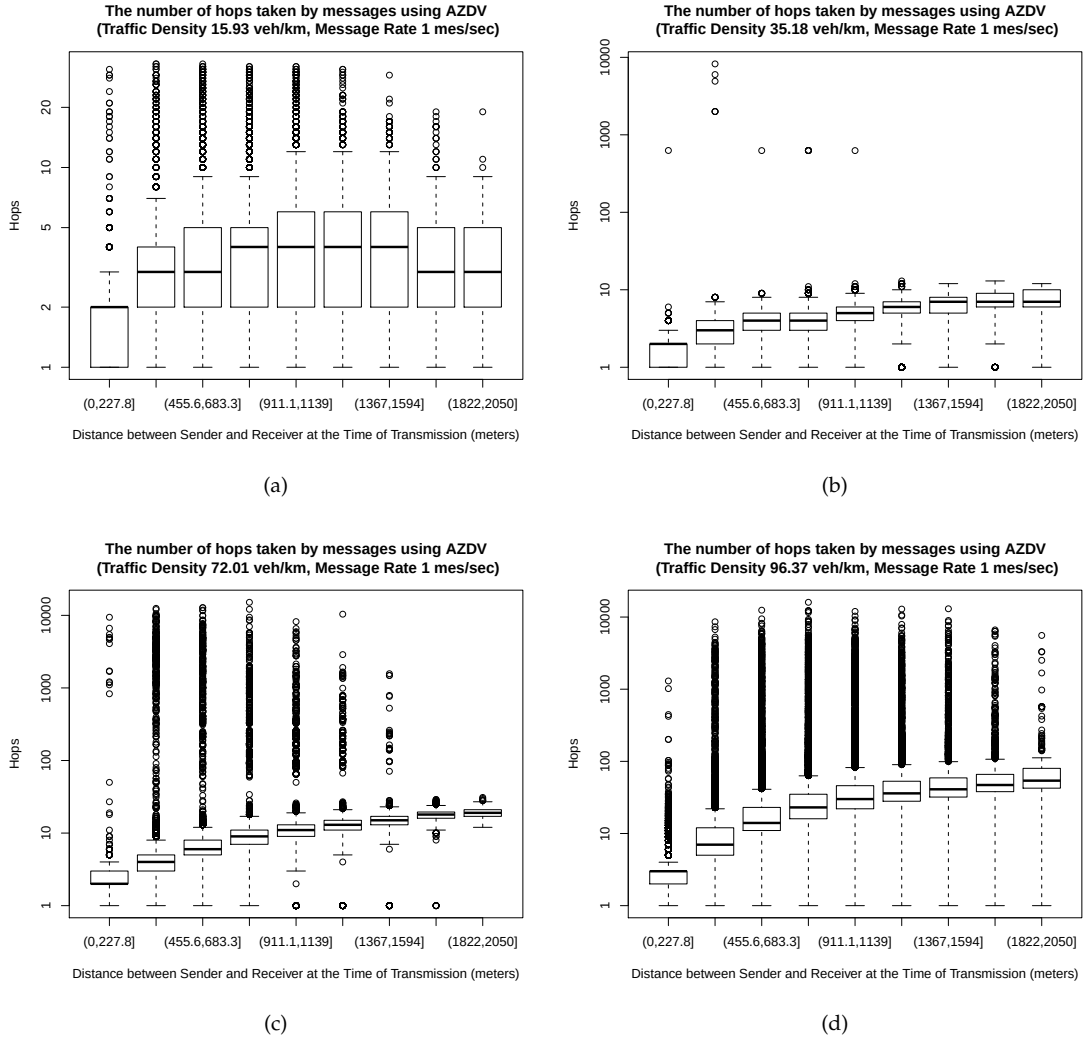
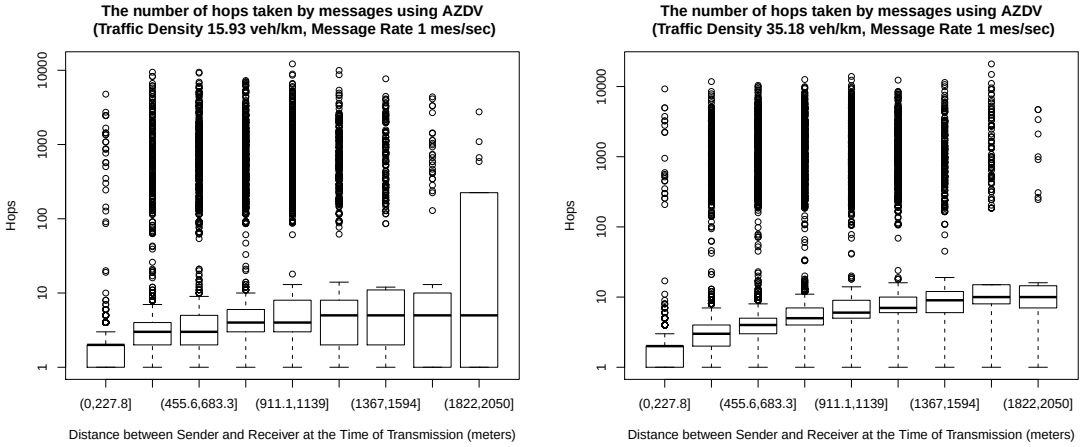
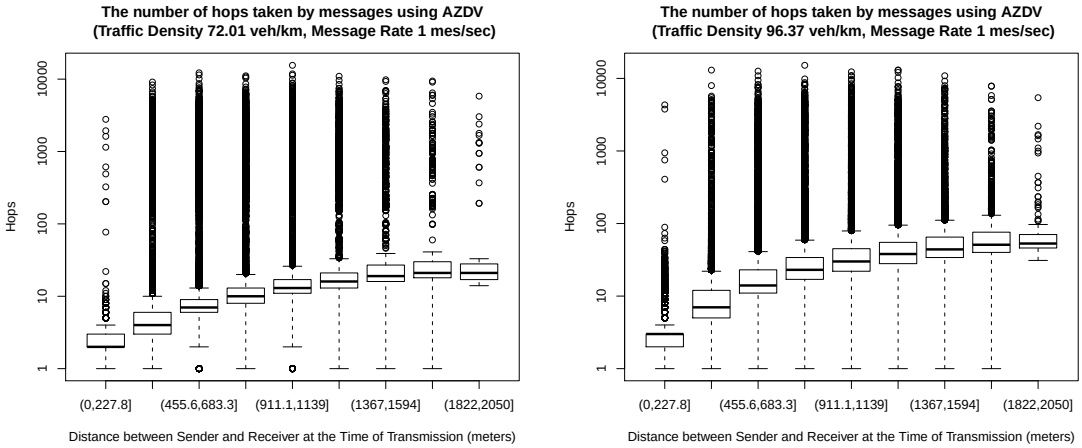


Figure 6.8: Exp2 AZDV with the max next-hop stack size set to 50 and carry-and-forward disabled: User Message Hops.



(a) (b)



(c) (d)

Figure 6.9: ExpB AZDV with the max next-hop stack size set to 50 and the carry-and-forward threshold set to 10 seconds: User Message Hops.

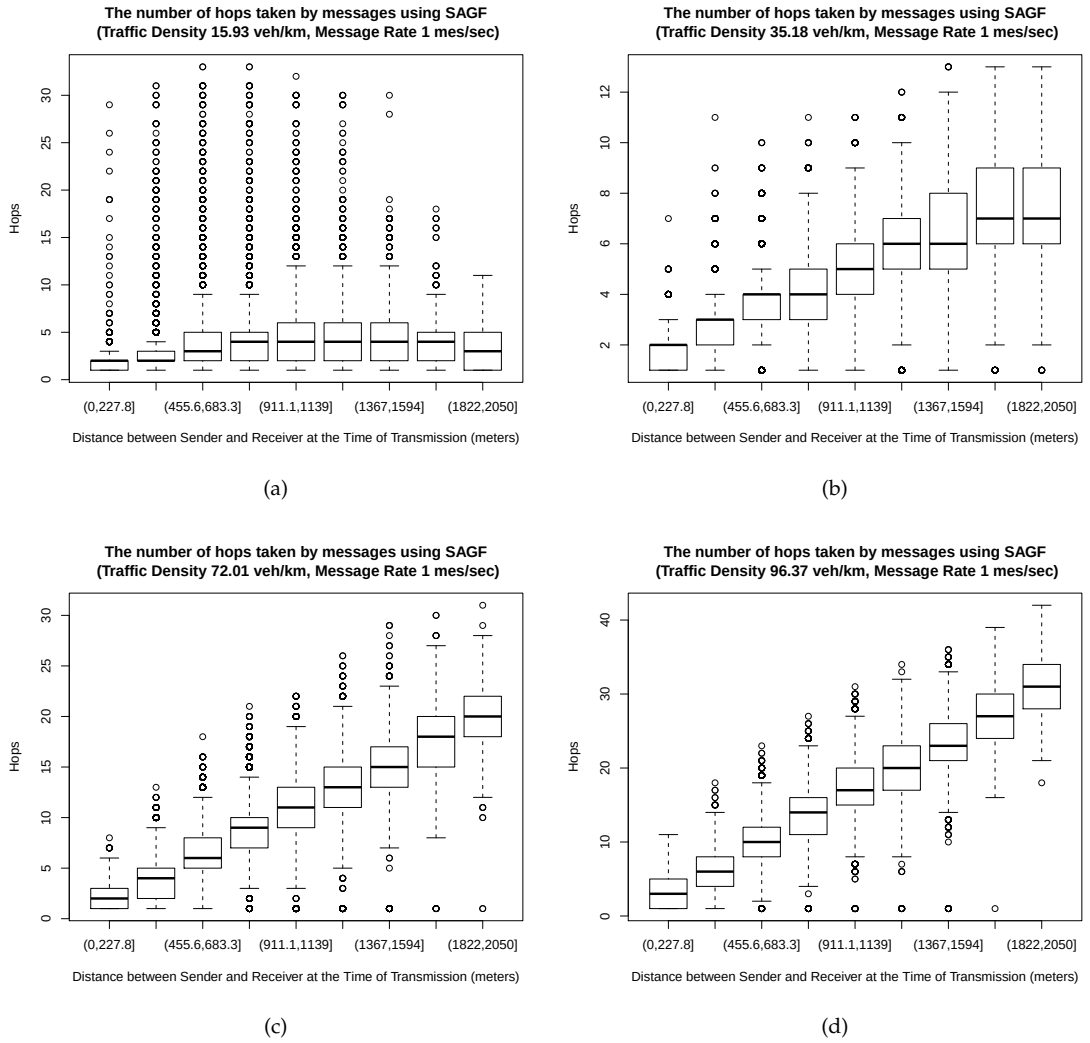


Figure 6.10: ExpB SAGF User Message Hops.

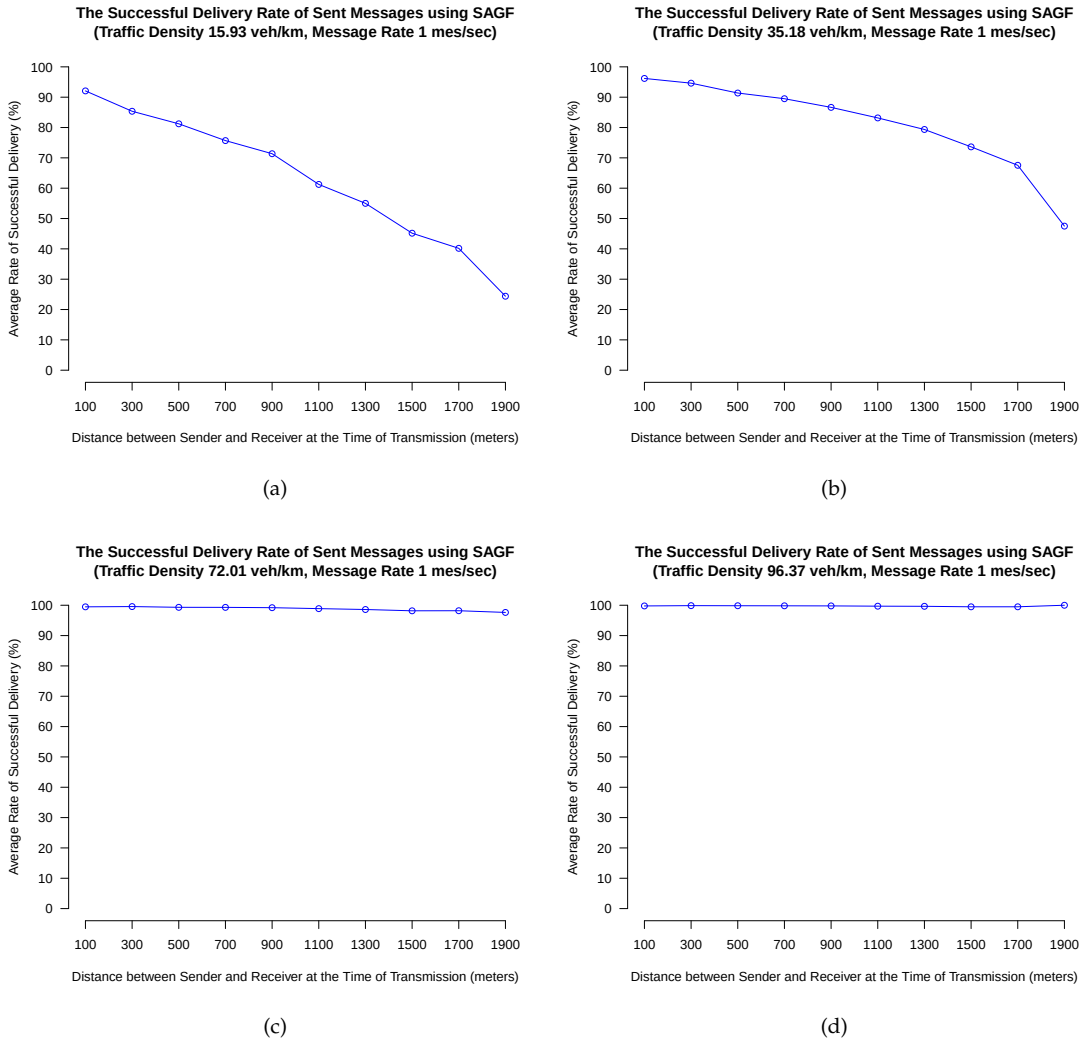


Figure 6.11: ExpB SAGF: Percentage of User Messages Delivered Successfully.

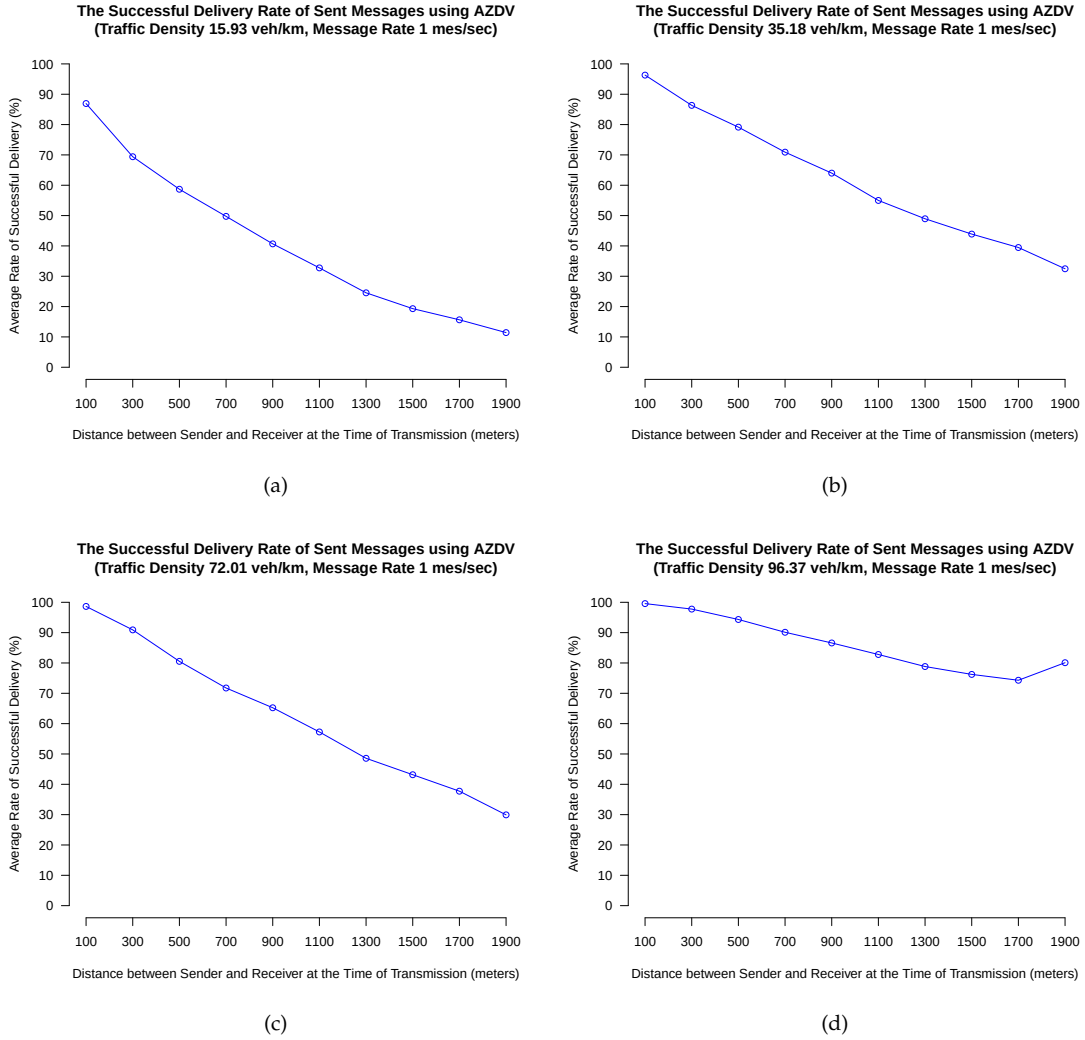


Figure 6.12: ExpB AZDV with a CaF Threshold of 10 seconds: Percentage of User Messages Delivered Successfully.

7 AZDV Partial Redesign

7.1 AZDV Routing Loops

The routing loops are a symptom of conflicting AZDV routing table records, between nodes in communication. The difficulty lies in isolating the cause of these AZDV record conflicts. Figure 7.1 shows an example of how, by a chain of next-hop entries, routes are formed in AZDV. In a system with full connectivity, an AZDV periodic routing message (Section 4.3.3) would propagate outwards and form a directed acyclic graph (DAG) of next-hop entries leading back to the origin of the message. However, AZDV operates in a partitioned network without full connectivity. It takes time for the AZDV routing records to propagate because the nodes in the network have to carry-and-forward the AZDV routing packets between unconnected zones.

In the design process, it was thought that node mobility would be helpful. By enabling carry-and-forward for AZDV routing packets, routes could be formed over areas of disconnection. However, node mobility can also create inconsistencies in AZDV routing information between nodes.

A major cause of the loops was discovered after studying the traffic traces and network statistics of the AZDV simulations affected worst by the loops. Figure 7.2 demonstrates a particular situation in which node mobility can create routing loops. Node A and B start on opposite borders of a zone and receive different next hops for zone A (a different zone to the one they inhabit). Node A and B are traveling in opposite directions and pass each other. Now, any packet addressed to zone A will become caught in a loop between node A and B. This was found to be the cause of the most severe routing loops. This does not necessarily occur between just two nodes either, the cycle can occur between two groups of nodes with conflicting AZDV next hops.

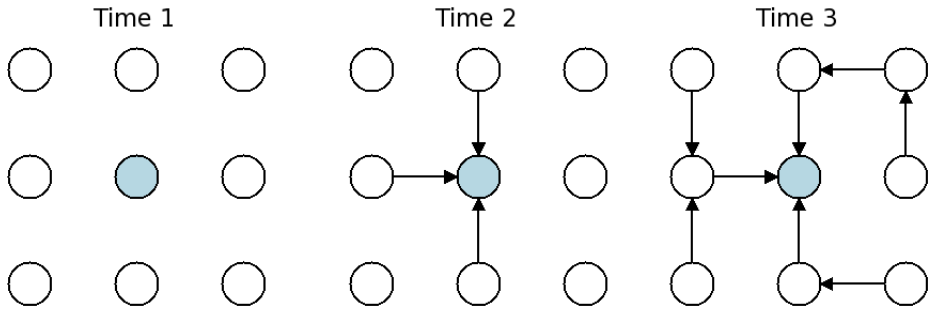


Figure 7.1: Representation of DAG Route Formation over time. The circles represent junctions in a grid road system. The arrows represent the next hop to reach the central zone from each zone.

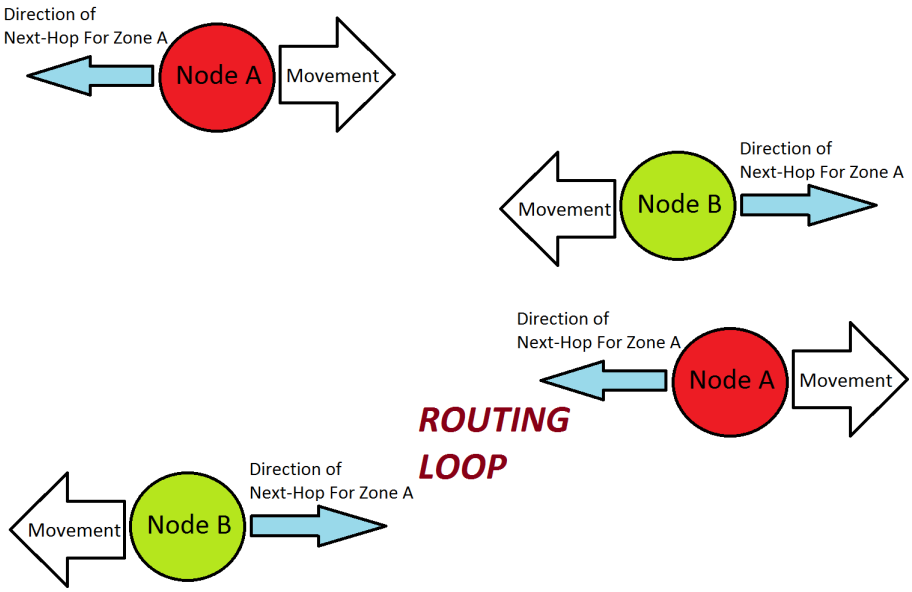


Figure 7.2: Both nodes have a different next-hop zone for the same zone. If their paths cross, their opposing next-hop directions create a routing loop. Any message sent to Zone A from either nodes will be caught in an endless cycle between the two nodes.

SAGF does not suffer from routing loops because SAGF has a deterministic next-hop zone for each route-able zone. However, SAGF is a stateless protocol so it is unable to adapt to varied traffic conditions, unlike AZDV. To solve the problem of routing loops in AZDV, both loop detection and loop elimination need to be introduced into the AZDV system. When a loop is detected, some form of action can then be taken to break the loop.

7.2 Loop Detection

There are two main candidates for loop detection in AZDV:

1. A previous forwarding node metric which records the last n nodes which have forwarded the packet. Any node attempting to forward a user message can check the forwarders list to see if it has forwarded this same message before. This is a node-space loop detection method which is ignorant of map-space in which AZDV operates.

2. A zone hop metric which records the last n junctions/zones the user message has been forwarded through. The list can be checked before forwarding to see if any of the previous zones in the position list are repeated. This is a higher-level map-space loop detection method.

Advantages of the *node identity* metric over the *zone hop* metric are that it is fine-grained enough to detect loops that occur within a single AZDV zone, which is where the majority of the routing loops occur. However, it is only effective on loops which contain fewer nodes than the

size of the previous forwarding node list. Unlike the *position* loop-detection metric, it is not able to detect large scale loops that may occur over long periods of time over larger areas.

With both metrics, bandwidth overhead needs to be taken into account. Each data packet sent through the network will need to maintain a queue containing the last n locations visited, each one of these entries will increase the bandwidth usage of the packet. Another option without the bandwidth overhead, would be for the nodes themselves to maintain a list of packets previously forwarded. The only down side to this approach is that packets are currently not uniquely identifiable.

Both methods of loop detection were implemented in a revised version of AZDV. Only the forwarding node loop detection metric was found to be effective. As a result, the redesigned AZDV only contains the forwarding node loop detection method.

7.3 Loop Elimination

AZDV is essentially a superset of SAGF. If a node does not have an AZDV routing record for the destination of a packet, it will default to SAGF behaviour. SAGF does not suffer from routing loops. The most obvious method of loop elimination is to force AZDV to revert to SAGF routing behaviour on messages that are found to be caught in forwarding loops.

In the new version of AZDV, upon detection of a routing loop in the current packet, the routing node will erase the AZDV record pertaining to the destination of the current packet. This will flush the conflicting routing information from every node that attempts to forward the looping packet twice. The packet will be forwarded onwards to the next zone by the deterministic greedy forwarding algorithm within AZDV (which is equivalent to SAGF).

8 Experimentation: Stage 2

This chapter documents the performance of AZDV with the modifications described in Chapter 7. Experiment A, which was the control experiment featuring the 5km linear road, will not be repeated in this section. It is sufficient to say that the modified AZDV performs as expected in experiment A, as it did in the previous round of experimentation in Chapter 6.

This chapter contains an analysis of the performance of the improved AZDV algorithm in experiment B. Experiment B is conducted on a realistic road topology. Experiment B is performed over two main categories of simulated traffic:

1. Uniform traffic traces (as a control)

2. More realistic non-uniform traffic traces

8.1 Experiment B with uniform traffic

The experiments performed in Section 6.2 were repeated for this section.

Uniform traffic traces were used as an experimental control in this run of experiment B. SAGF is a stateless routing algorithm which makes it unable to adapt to traffic conditions. However, the routing decisions of SAGF are optimal in uniform traffic.

SAGF uses Dijkstra's shortest path algorithm (Dijkstra, 1959) to calculate the shortest routing paths. The cost metric used is Euclidean distance over the road network. In a uniform traffic trace, the traffic parameters for speed and density are constant across the road network. In chapter 6, it was shown in Figure 6.2, that in uniform traffic, the message delivery delay using SAGF is linearly proportional to the distance from the source and destination of the message. Therefore SAGF correctly calculates the probable fastest routes for uniform traffic.

The performance of AZDV can be best judged in this experiment by observing how close it matches the performance of SAGF.

8.1.1 Experimental Parameters

The following is the list of experiment parameters used, which follow the structure in 5.4:

- Road Map Square Grid Dimensions (km^2) = 1

- Road Map Number of vertical and horizontal roads = 6 (The grid is composed of road

segments of 200m in length)

- Traffic Type = {uniform, non-uniform (The traffic density of each straight road section is set to a random value between 0 and the Traffic Density test parameter) }
- Average Traffic Density (vehicles/km section of road) = uniform density{ 15.93, 35.18, 72.01, 96.37}, variable density{ 15.93, 55.09, 111.12, 176.87 }
- Vehicle Max Speed (miles/hour) = 70
- VANET Penetration Rate = 1.0
- Max Wireless Channel Capacity = 3 Mbit
- Max Wireless Range = 50
- Routing Methods = { SAGF, AZDV }
- Message send rate (Hz)= { 0,1,2,4,8,16 }
- AZDV Zone Data Refresh Rate (Hz)= 1
- AZDV Carry-and-Forward Threshold (seconds) = { 10, ∞(equivalent to disabling carry-and-forward) }
- AZDV Max 'Next Hop' Stack Size = { 50 }

The simulation was run for a total of 400 simulated seconds. The first 100 seconds are unmeasured by the metrics, the purpose of the initial time period is to allow the AZDV routing packets to propagate through the system. All user messages created between experiment time 100-150 are measured by the experiment metrics until the experiment ends at experimental time 400. User messages created after time 150 are not measured by the metrics but still contribute to the simulated load on the network while the measured messages move towards their destinations.

8.1.2 Results

Figure 8.1 shows the bandwidth usage per node of the revised AZDV system in comparison to SAGF. The bandwidth usage of AZDV seems to scale better with respect to SAGF than it did before it had routing loop correction measures in Figure 6.7. However, with both Carry-and-Forward enabled and disabled, the bandwidth overhead of the simulation with the highest traffic density is still not ideal.

If the AZDV routing system was functioning perfectly, the bandwidth overhead of AZDV would be constant across all message send rates. In a completely idealized scenario, AZDV bandwidth usage would be linearly proportional to the number of route-able zones in the system. it would be invariant to traffic density. However, as Figure 8.1 shows, under the highest traffic density simulation, the AZDV bandwidth overhead rises with the message send rate. This could indicate that some routing loops are still occurring.

Figure 8.2 shows the average hop counts for AZDV. It is a definite improvement over the previous performance of AZDV in Figure 6.9. The maximum hop count recorded is around 2 orders of magnitude less for the revised AZDV algorithm over the original AZDV algorithm. In the original version of AZDV, hop counts were recorded in excess of 10000, now the maximum recorded are only around 200 hops.

However, the hop counts still seem a little higher than they should be. For instance, for the highest traffic density, the recorded hop counts for messages which are sent a distance of under 227.8 meters can take in excess of 50 hops. Compare this to the recorded hop counts of SAGF for the same distance in Figure 6.10, no message takes in excess of 12 hops for the same distance.

SAGF is the gold standard in uniform traffic because its shortest path algorithm uses the metric of physical distance, which is an accurate metric when the traffic density is uniform. The elevated hop counts indicate the possibility that some routing loops are still occurring in the revised AZDV algorithm. That said, the new loop-detection measures seem to be eliminating the more severe cases.

The previous version of AZDV did not produce message delivery success rates that were competitive with SAGF, which was illustrated in Figure 6.11 and Figure 6.12. Figure 8.3 displays the improved performance of the revised AZDV algorithm which manages a similar proportion of successful message deliveries to SAGF. Both SAGF and AZDV now attain close to 100% delivery rates for the two higher traffic densities.

Figures 8.5 and 8.4 show AZDV and SAGF to be very close in performance. Both have similar median values for delivery delay. The differentiating characteristic of AZDV over SAGF is its higher spread, there are some outliers which have double the maximum delivery delay of SAGF.

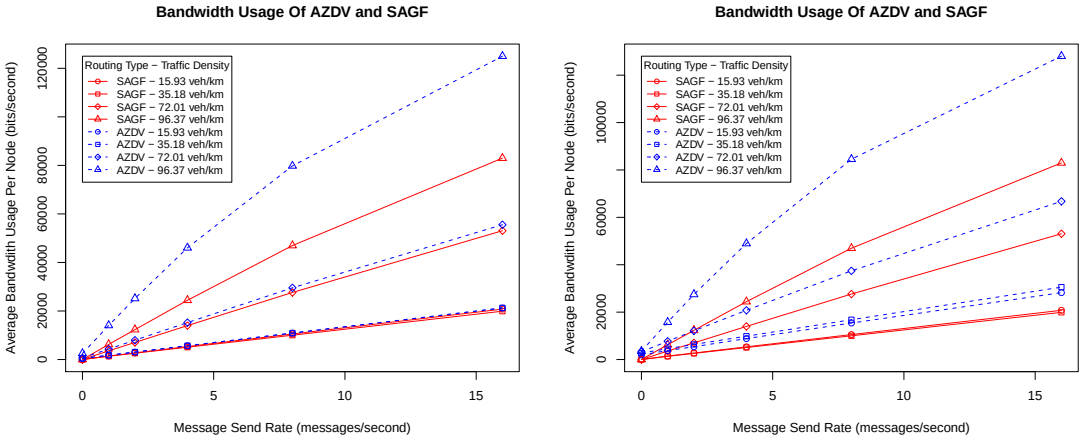
The revised AZDV algorithm now attains similar performance to SAGF in the key areas of message delivery delay and message delivery success. Areas for concern with the revised AZDV include the elevated bandwidth overhead it incurs, which indicates the presence of routing loops, albeit less severe than the loops that occurred in the previous version of AZDV. For lower user message send rates, the bandwidth overhead does not seem to impair the performance of AZDV in other areas.

8.2 Experiment B on a Non-Uniform Traffic Trace

This run of experiment B used a non-uniform traffic trace designed to mimic real traffic conditions. SAGF essentially assumes uniform traffic when making routing decisions. AZDV adapts to real-time traffic conditions by propagating AZDV routing packets. This experiment was designed to show the ability of AZDV to adapt to variable traffic conditions and outperform SAGF.

8.2.1 Results

In the previous section, AZDV was shown to have similar performance to SAGF in the areas of message delivery delay and message delivery success. This was for uniform traffic simulations



(a) Carry-and-Forward Threshold = ∞ seconds

(b) Carry-and-Forward Threshold = 10 seconds

Figure 8.1: Exp B SAGF and AZDV (with anti-routing-loop modifications) Bandwidth Usage

where the traffic is the same density and speed on all roads. This gave SAGF the advantage because in a uniform traffic density, the shortest path for forwarding a message would be the same as the shortest physical distance, which is the metric by which the SAGF shortest path functions. AZDV routing packets would be discovering routes that SAGF already uses, incurring an unnecessary bandwidth overhead.

Real traffic is not uniform across entire road networks, it varies in density and speed. To simulate this, a non-uniform traffic trace was created for the grid map used in experiment B. The speed of traffic along each strip of road in the grid was set randomly to a maximum speed between 20 and 70 mph. The rate of generation of traffic along each of these strips was varied between 0 and the rate required to sustain the desired traffic density. Traffic density can vary between very sparse and complete deadlock.

AZDV should now have an advantage over SAGF. Assuming it functions according to design, it will be able to generate faster packet routes than those used by SAGF. Figure 8.6 shows the bandwidth usage of AZDV and SAGF on a simulated non-uniform traffic trace. The first thing to notice is that the bandwidth overhead of AZDV is high, enough to run into the 3Mbit bandwidth cap as demonstrated by the curve of the graph. The routing loops are burdening the system more severely for these non-uniform traffic traces.

Figure 8.9 and Figure 8.10 show the message delivery delay of SAGF and AZDV respectively. Despite the routing loop related high bandwidth overhead of AZDV, it outperforms SAGF in message delivery delay for the three highest traffic density simulations. The median message delivery delays for AZDV are as low as a tenth of those for SAGF in the higher density simulations. The spread of AZDV is also lower, with SAGF producing much longer message delivery delays outliers than AZDV. These results confirm that AZDV is able to adapt routing paths to real-time traffic conditions. The dramatically low message delivery delays can be attributed to the occurrence of high density traffic along some road segments. These roads feature strong end-to-end wireless connectivity. SAGF is unable to identify these low delay routes because it is a stateless routing algorithm.

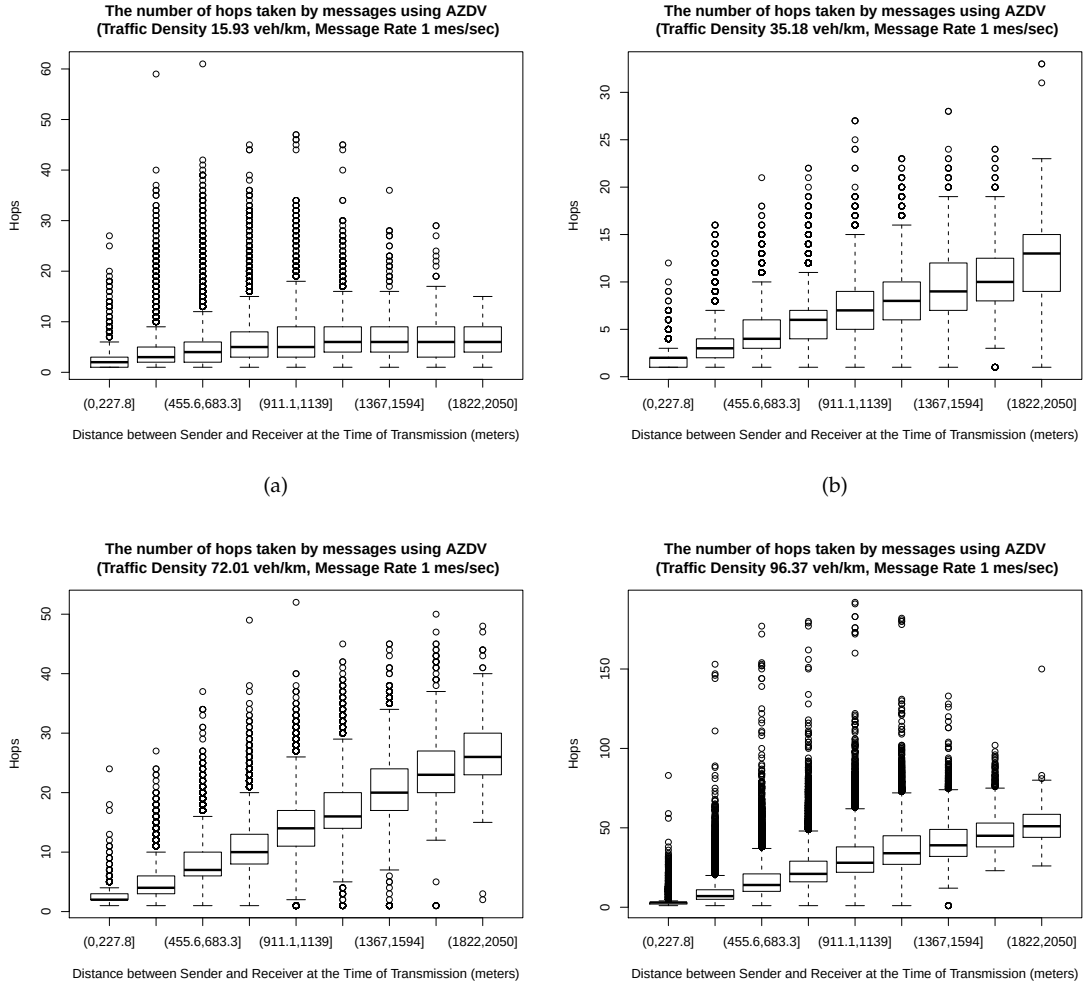


Figure 8.2: Exp B AZDV (with anti-routing-loop modifications): Average Hops per User Message.

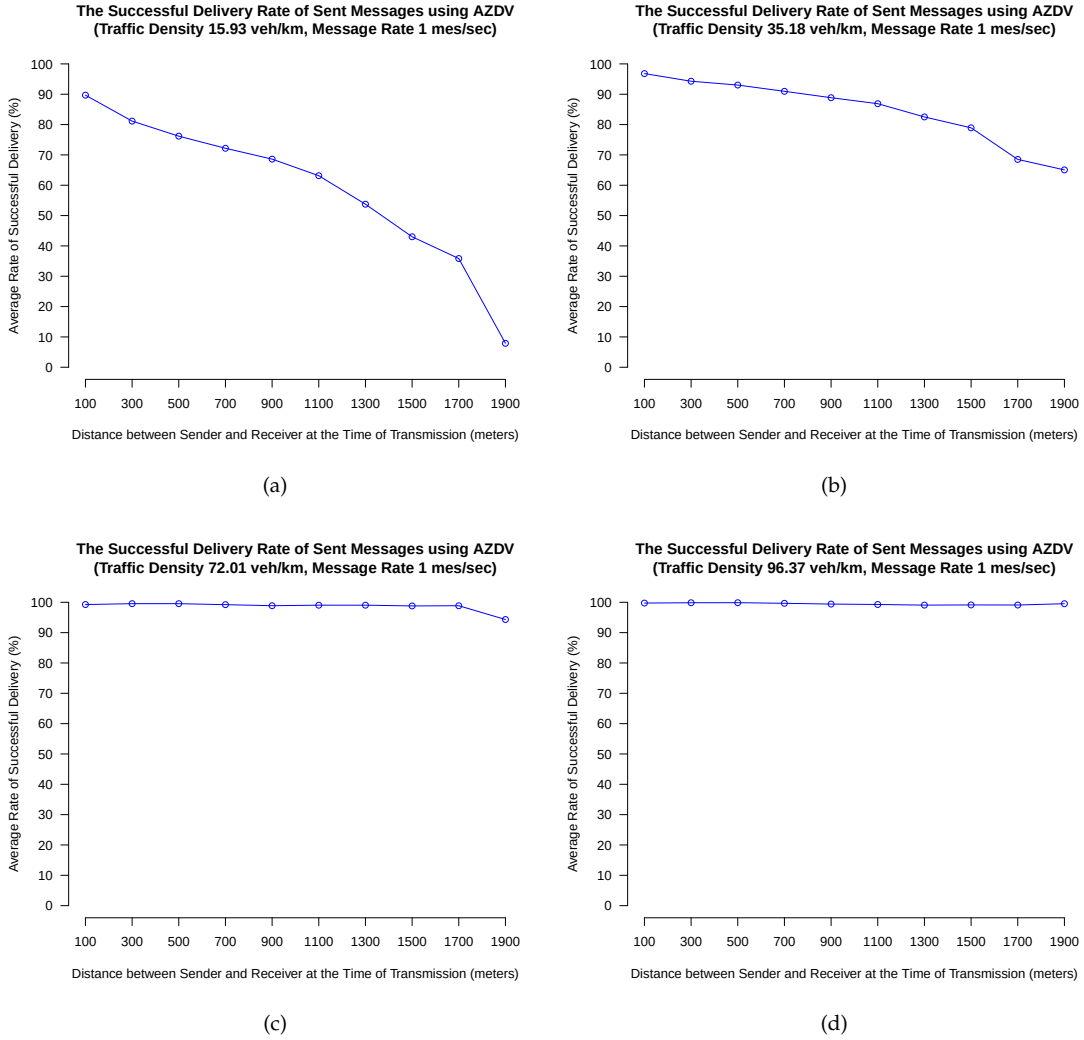


Figure 8.3: Exp B AZDV (with anti-routing-loop modifications): Proportion of successfully delivered User Messages.

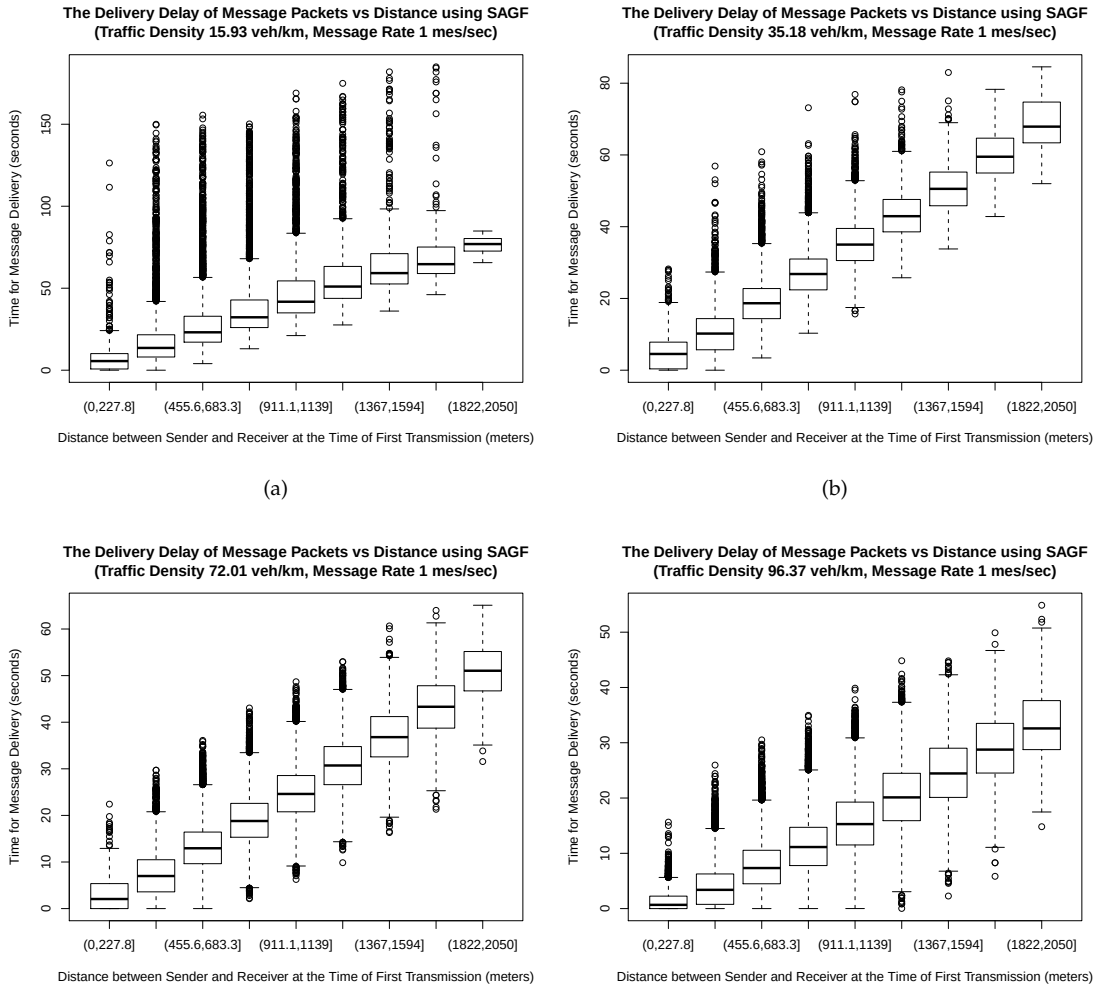


Figure 8.4: ExpB SAGF: User Message Delivery Delay.

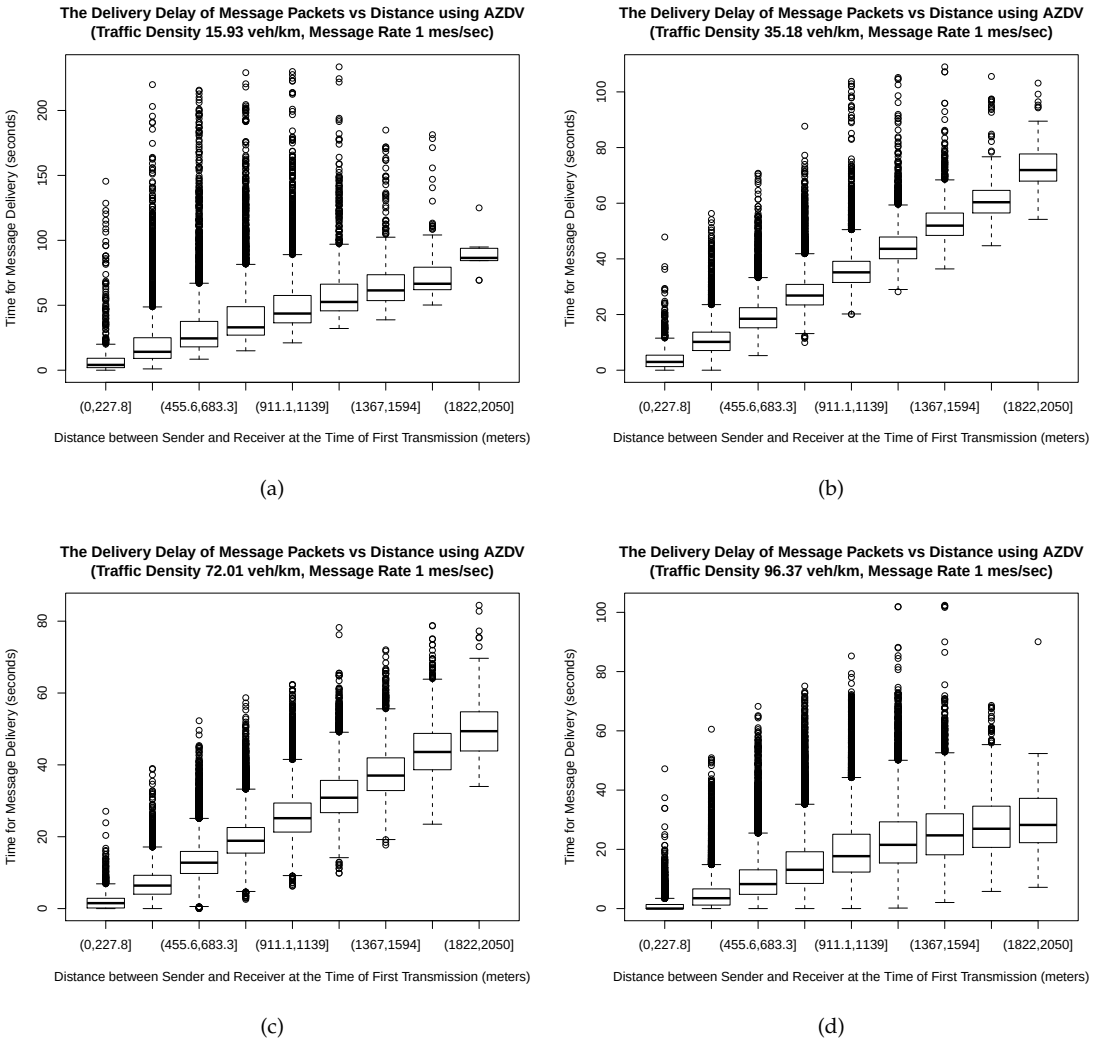


Figure 8.5: Exp B AZDV (with anti-routing-loop modifications): User Message Delivery Delay.

Figure 8.11 and Figure 8.12 show the proportion of messages that are successfully delivered by SAGF and AZDV respectively. AZDV also outperforms SAGF in this metric. AZDV consistently attains over 95% successful delivery rates for the three higher traffic density simulations. The reason for the poor performance of SAGF in this metric is largely because a proportion of messages were so slow in transit that the simulation ends before they reach their target zone. The messages are undelivered as a result. This metric is dependent on the delivery delay metric.

Figure 8.7 and Figure 8.8 show the hop counts of SAGF and AZDV respectively. Both figures look similar to their counterparts for the uniform traffic trace. SAGF shows consistent hop counts that seem to vary linearly in proportion to the distance a message is sent. The median hop count of AZDV seems to follow this linear pattern also. However, the hop counts of AZDV have a large spread, featuring large amounts of outliers. It is these outliers that would seem to indicate the presence of routing loops. The lowest density traffic trace seems to lack the dense spread of outliers and has a correspondingly low bandwidth overhead in Figure 8.6.

8.3 Evaluation

This experiment has demonstrated the ability of AZDV to successfully adapt to real-time traffic conditions and minimize packet delivery delays. In the experiment with the non-uniform traffic trace, the results of AZDV are very impressive. These were two out of three of the design goals defined in 3. However, in its revised state, AZDV still suffers from routing loops which are not trivial to eliminate. Further work needs to be performed to eliminate routing loops in order to satisfy the final design goal which is to minimize bandwidth usage.

The current AZDV routing loop prevention measures worked to a certain extent. The most severe routing loops were eliminated. The eliminated routing loops occurred under continuous wireless connectivity between all nodes in the loop, which accounted for the incredibly high hop counts in the short simulation time. This made these loops easy to detect by adding a list of previous forwarding nodes to each transmitted user message. The loops which couldn't be detected were those that:

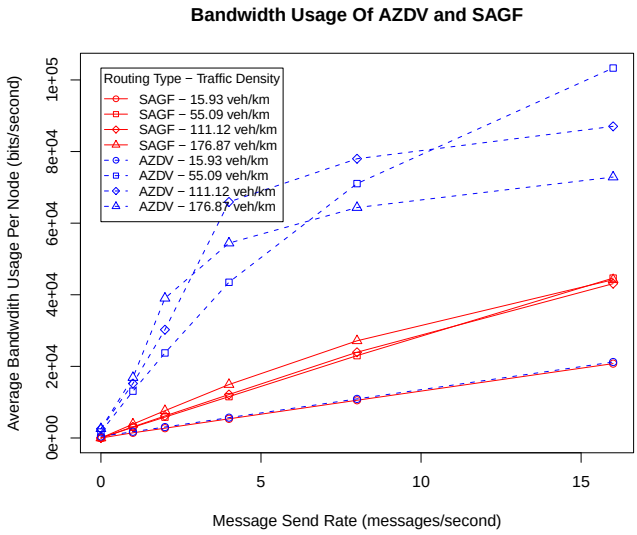
1. Occurred over longer periods of time and that did not contain duplicate nodes in the loop. A loop in terms of repeatedly visiting the same zones rather than repeatedly forwarded by the same node/s.

2. Loops which featured more nodes in the loop than the max size of the previous forwarders list in the data packet (in simulations, this size was 10).

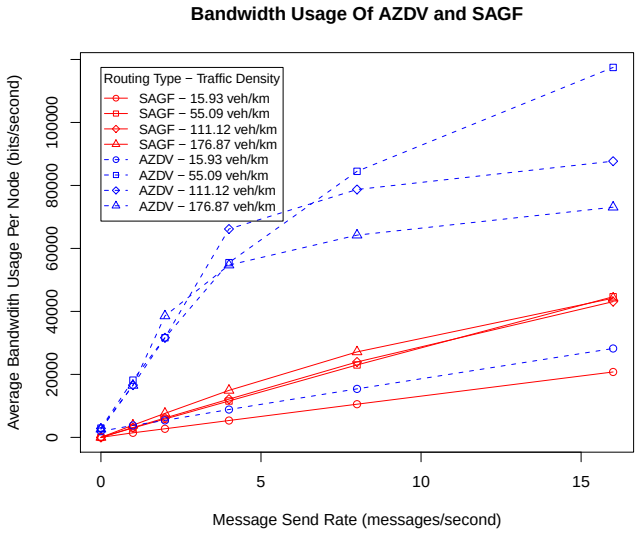
As a second measure of loop detection, another list was added to each AZDV record which recorded each previously visited zone. However, the results when using this routing loop detection metric were poor, so it was eliminated.

Routing loops were not originally considered in the design of AZDV. When they were discovered, AZDV was retro-fitted to eliminate the routing loops. The implementation time for this feature was extremely limited due to limited time frame of the project. The method used to eliminate detected loops was simplistic, consisting of simply erasing the parts of the AZDV table which were discovered to correspond to loops. This reverted the AZDV behaviour back to that of SAGF temporarily for any user messages with detected loops. This method proved successful but did not completely remove the loops. Any further work on AZDV should concentrate on

eliminating routing loops.



(a) Carry-and-Forward Threshold = ∞ seconds



(b) Carry-and-Forward Threshold = 10 seconds

Figure 8.6: Exp B (with anti-routing-loop modifications) Bandwidth Usage

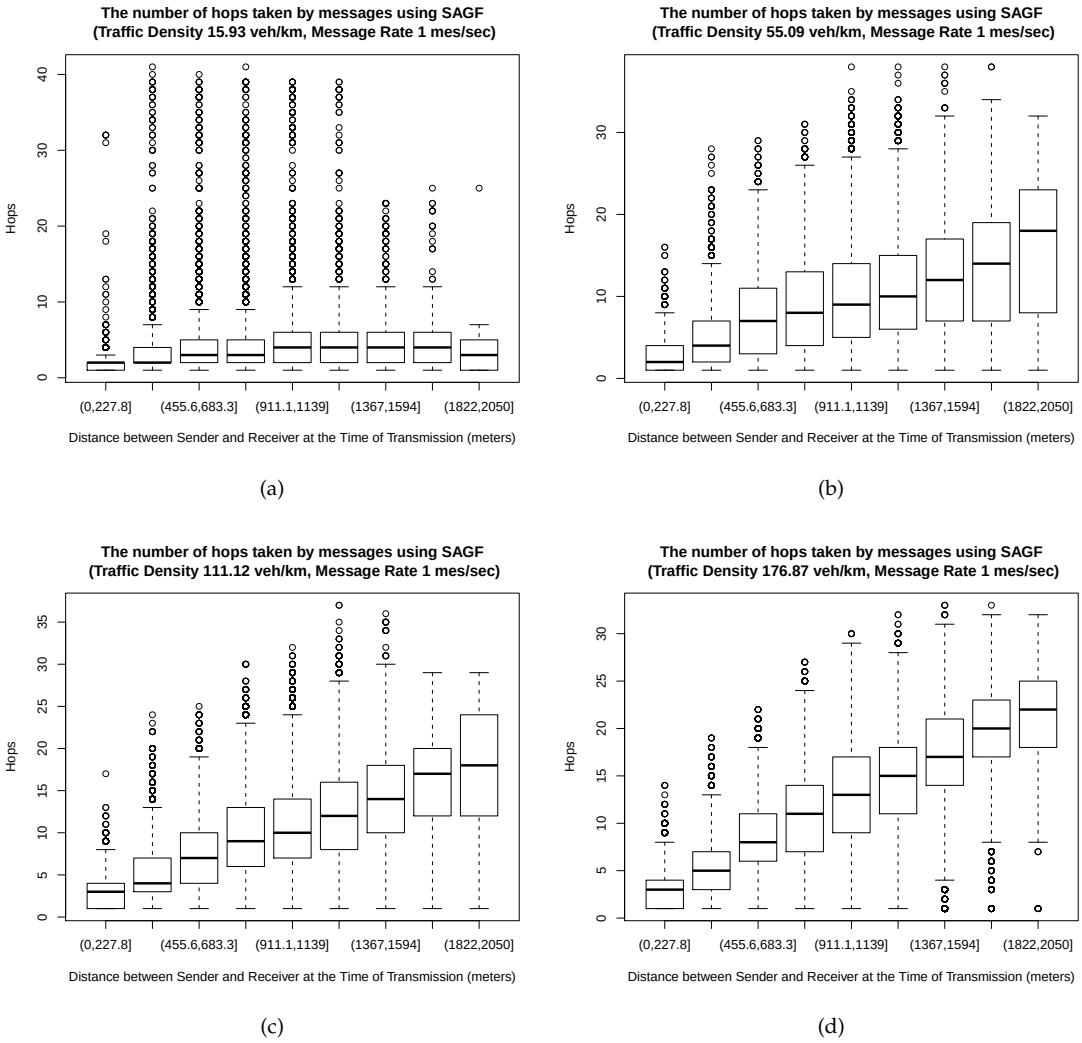


Figure 8.7: Exp B SAGF: Average Hops per User Message.

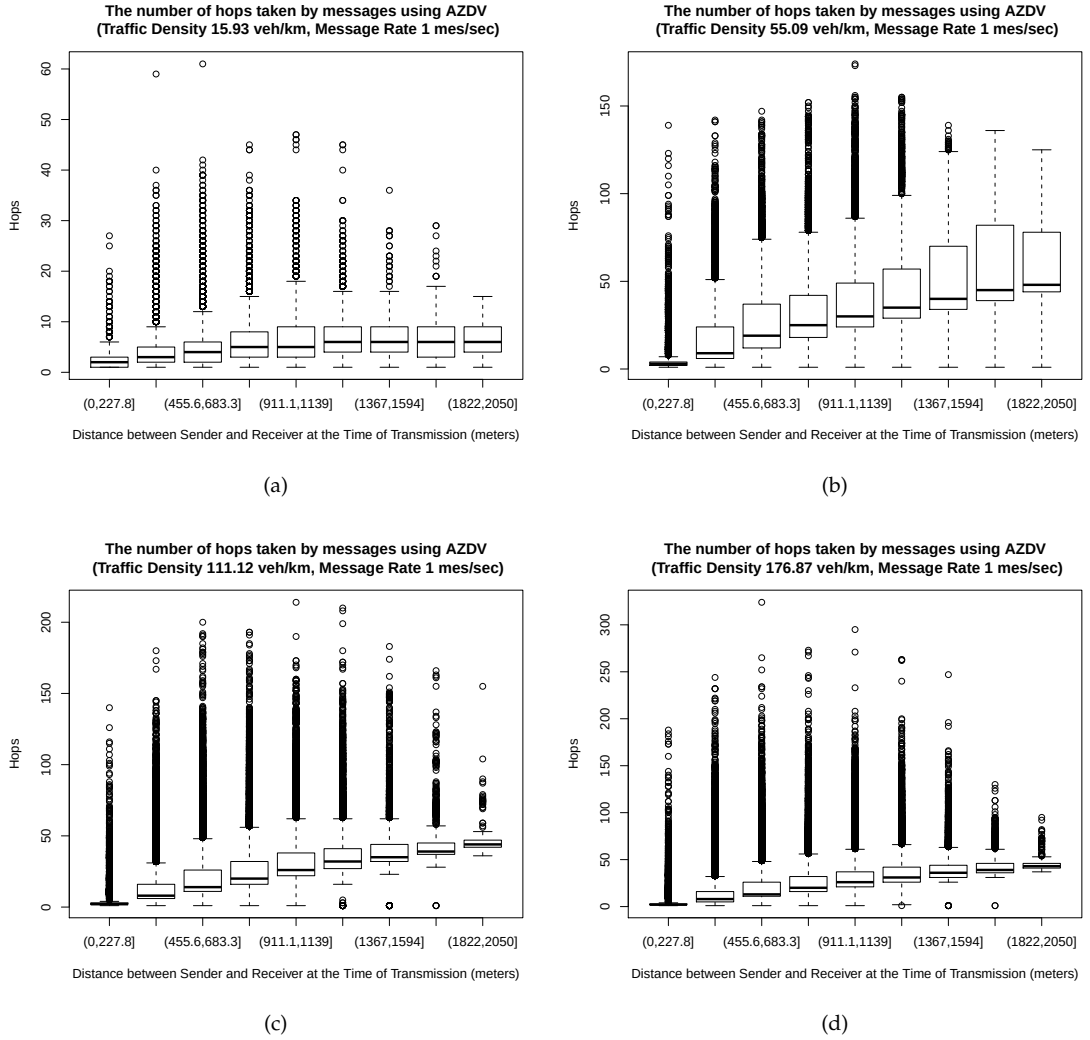


Figure 8.8: Exp B AZDV (with anti-routing-loop modifications): Average Hops per User Message.

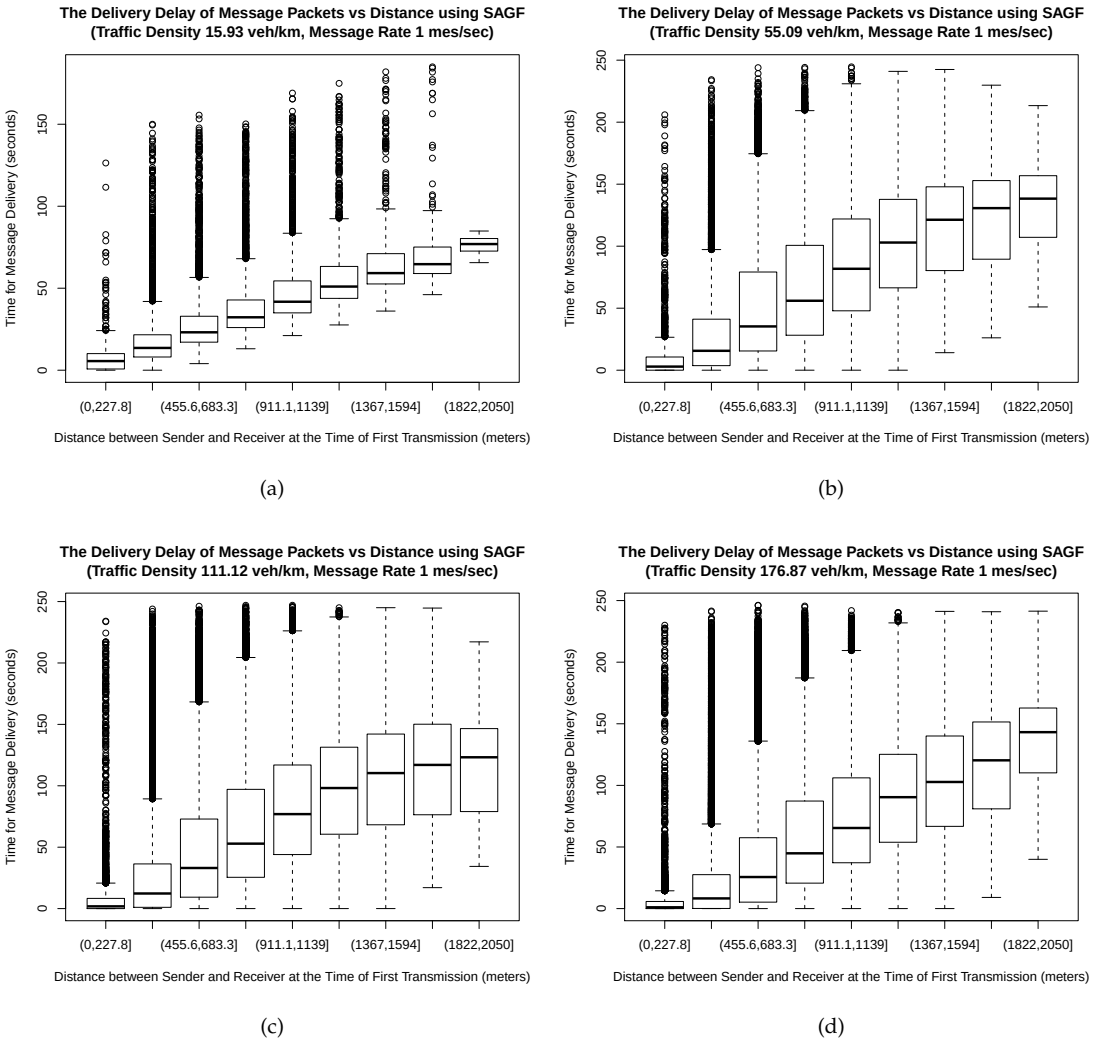


Figure 8.9: Exp B SAGF: Average Delivery Delay per User Message.

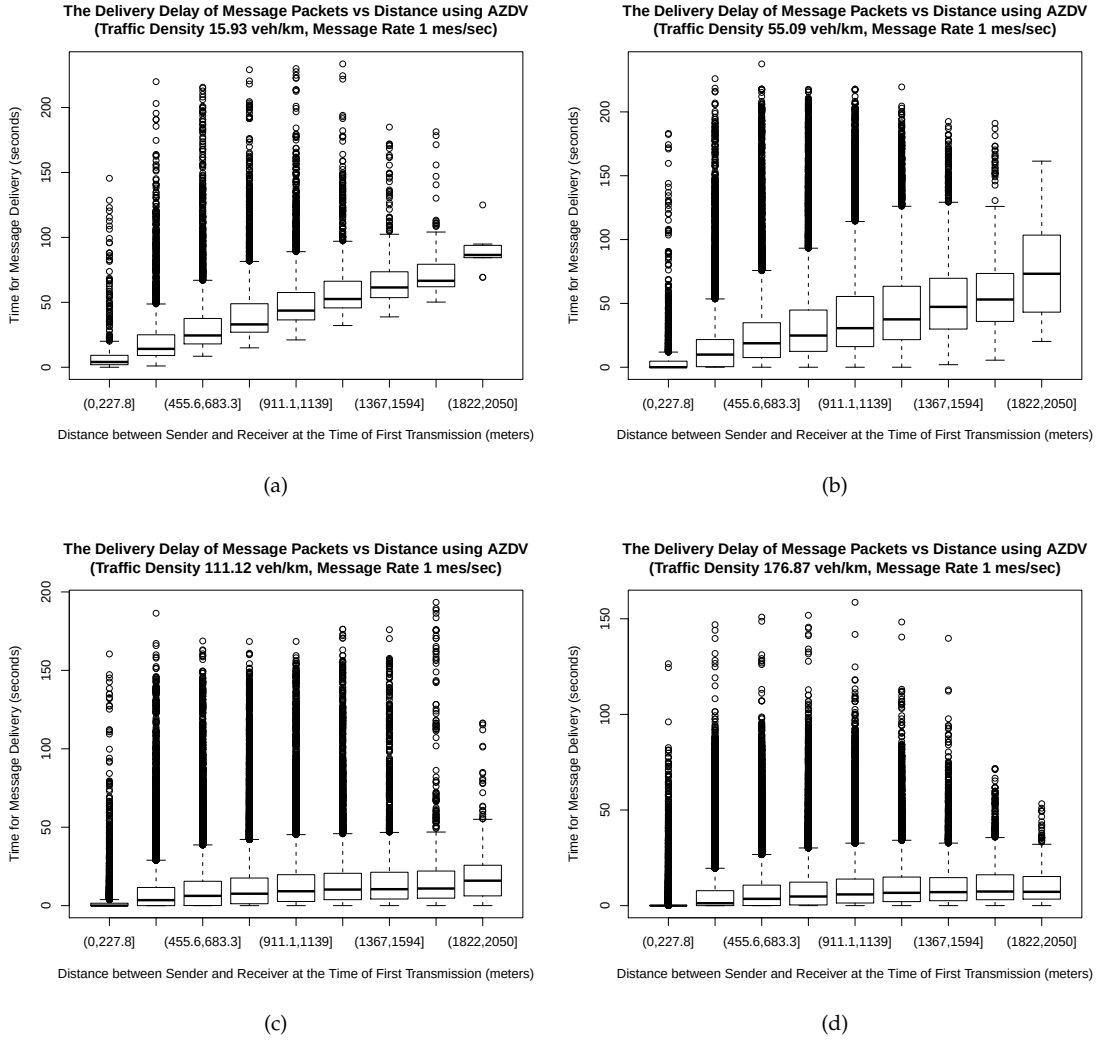


Figure 8.10: Exp B AZDV (with anti-routing-loop modifications): Average Delivery Delay per User Message.

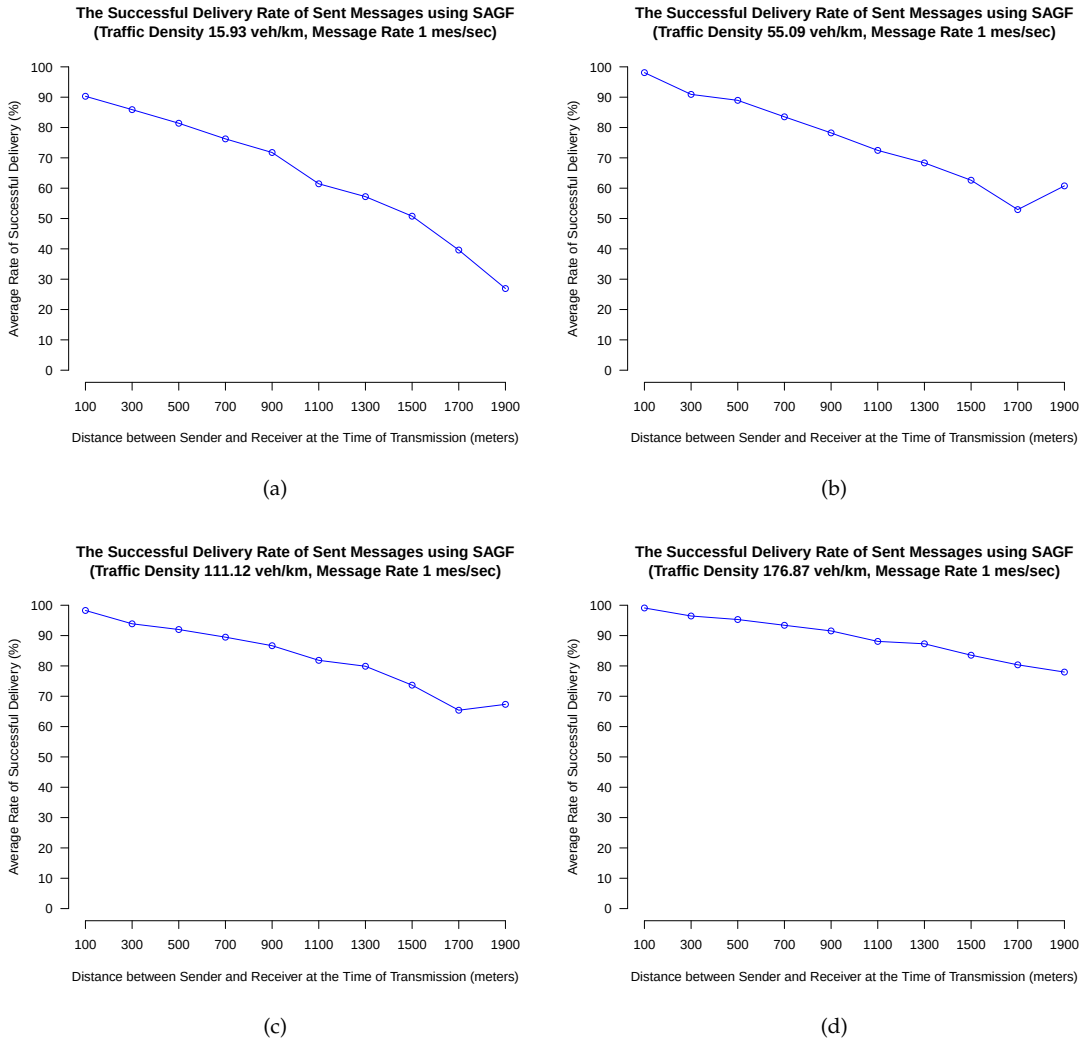


Figure 8.11: Exp B SAGF: Proportion of successfully delivered User Messages.

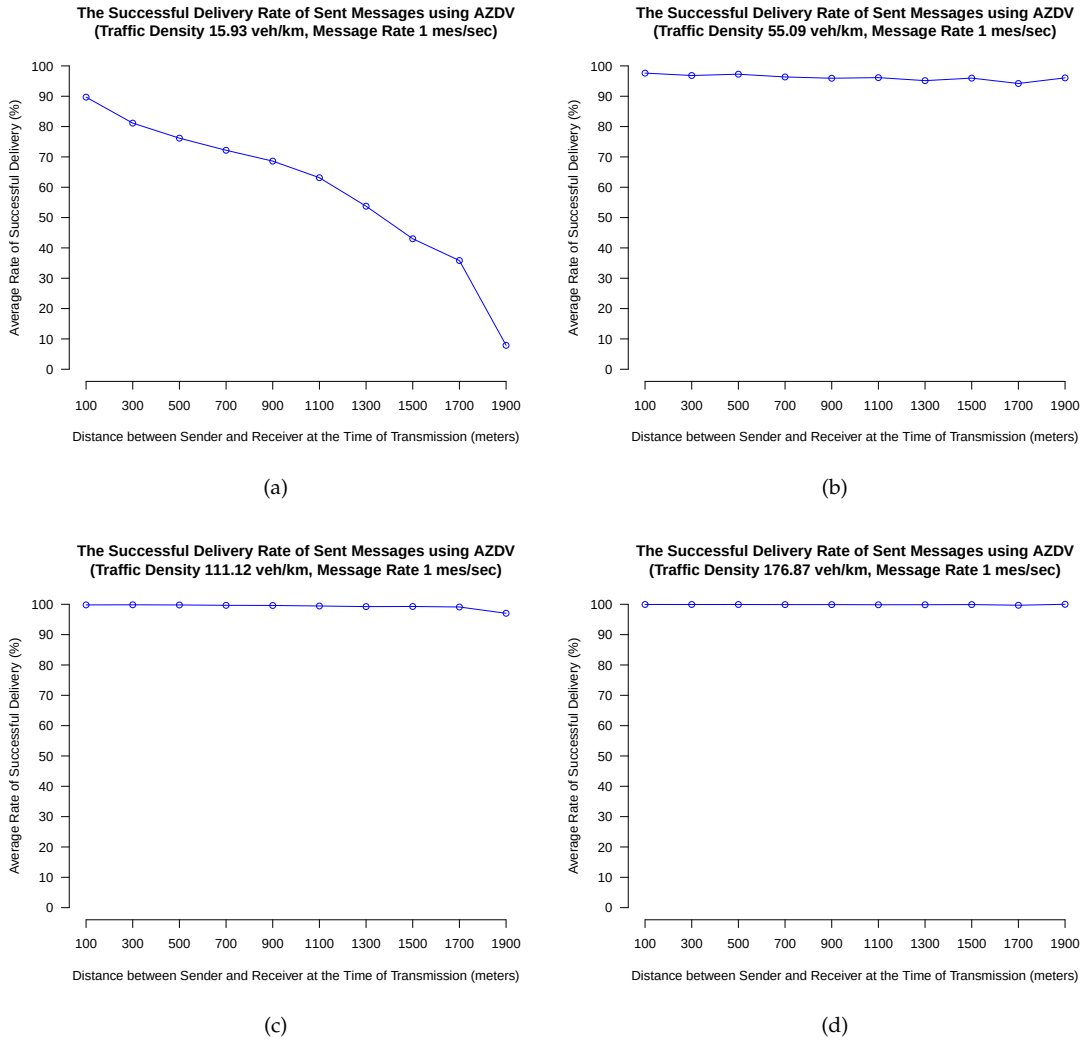


Figure 8.12: Exp B AZDV (with anti-routing-loop modifications): Proportion of successfully delivered User Messages.

9 Evaluation

9.1 Overview of Experimental Results

AZDV is a VANET routing algorithm designed by this author and presented in this thesis. It was designed to fulfill four main design goals introduced in Chapter 3:

1. Minimize packet delivery delay
2. Improve packet delivery reliability
3. Adapt to real-time traffic conditions and network loads
4. Minimize the bandwidth overhead of the routing algorithm

In order the evaluate the performance of AZDV according to these goals, an existing VANET routing algorithm known as spatial aware geographic forwarding (SAGF) (Tian et al., 2002) was evaluated alongside AZDV. All experimentation was performed on both algorithms to evaluate the performance of AZDV relative to SAGF as a benchmark.

Test scenarios with both uniform and non-uniform traffic traces were simulated. AZDV excelled in the non-uniform traffic traces, while SAGF had a slight performance advantage in the uniform traffic simulations. The non-uniform traffic trace is a better approximation of real traffic conditions than the uniform traffic trace.

SAGF excelled in the uniform trace due to several key aspects of its design:

1. SAGF is a stateless routing algorithm unlike AZDV. AZDV incurs bandwidth overheads with the maintenance of its *distance vector* routing tables. SAGF uses less bandwidth than AZDV when both are used in identical circumstances.
2. To calculate routing paths, SAGF uses Dijkstra's shortest path algorithm (Dijkstra, 1959) with a cost metric of Euclidean distance. In uniform traffic, the speed and density of traffic is uniform across all roads. Delivery times of messages sent through the network are linearly proportional to the physical distance they have to travel which was demonstrated experimentally and presented in Figure 6.2. For simulations using uniform traffic traces, the routing algorithm of SAGF is actually very accurate despite being unable to measure the state of the network.

Despite the advantages of SAGF in uniform traffic, AZDV attained similar performance in the metrics of message delivery delay and message delivery success. The only metric showing large discrepancies between SAGF and AZDV was that of bandwidth consumption. SAGF consistently used less bandwidth than AZDV because of its stateless nature.

Non-uniform traffic simulations are a much closer approximation to real traffic conditions. In a non-uniform traffic scenario, AZDV was shown to have superior performance over SAGF in terms of minimized packet delivery delays and packet delivery success. The large improvement of packet delivery delay of AZDV over SAGF highlighted the ability of AZDV to adapt to real-time traffic states. In some cases AZDV delivery delays were a tenth of those produced by SAGF. Such dramatic improvements can offset the additional bandwidth overheads that AZDV incurs.

9.2 Conclusion

9.2.1 Map Overlay Routing

AZDV is a map overlay routing algorithm. It is not unique in this respect. However, it is worth reviewing the advantages it provides over MANET routing algorithms such as GPSR (Karp and Kung, 2000). AZDV has demonstrated these benefits in experimentation.

It is able to utilize knowledge of the road network topology to better navigate around topology holes such as buildings (Lochert et al., 2005). It is able to better calculate routes by applying Dijkstra's shortest path algorithm to a graph representation of the road network using a cost metric (Tian et al., 2002). Map overlay routing also offers predictable scalability compared to node space routing techniques. The data overhead of AZDV scales linearly with the number of route-able zones in the VANET system. Routing zones are pre-determined and fixed which makes them a valuable network abstraction.

9.2.2 Novel Techniques

AZDV introduces two novel techniques to the field of VANET routing.

AZDV successfully demonstrates that a distance vector (DV) algorithm can be successfully applied to VANETs with positive results. By designing AZDV to operate on a map overlay layer, it was able to overcome difficulties which would have plagued previous DV algorithms operating in a VANET/MANET, such as high node mobility and network fragmentation. AZDV, like DSDV (Perkins and Bhagwat, 1994), is not susceptible to the count-to-infinity problem. In experimentation, the DV routing approach was shown to minimize packet delivery delays in comparison to SAGF.

AZDV successfully demonstrates the direct measurement of packet delivery delays within a VANET network. It performs these measurements for the purpose of improving routing decisions compared to existing VANET routing techniques. Packet delivery delay is used as the main routing cost metric in AZDV. This gives AZDV an advantage over VANET routing algorithms that use Euclidean distance as the main routing metric i.e. SAGF (Tian et al., 2002), LOUVRE (Lee et al., 2008). VADD (Zhao and Cao, 2006) uses an estimated packet delivery delay as the main routing cost metric so it is not able to react to real-time conditions in the VANET communication network, such as network congestion.

9.2.3 Future Work

Overall, the experimental results for AZDV are promising. It offers superior routing performance to SAGF which justifies further work. However, AZDV has a number of limitations which require further research.

The AZDV bandwidth usage overheads were much higher than expected. It is clear from the experimental results that this overhead can be attributed to data packet routing loops. The bandwidth overhead of the AZDV routing packets is quite small and is observable directly when the system is under zero load. When a load is added to the AZDV system (by generating user data packets), the bandwidth overhead of AZDV in comparison to SAGF increases. The extra overhead is from an elevated number of user data packet hops in the system which can be attributed to routing loops.

When a data packet is stuck in a routing loop it is forwarded through the same set of zones/nodes repeatedly. Data bandwidth is wasted every time a packet is re-forwarded unnecessarily. Eventually, either the user message will be lost in a continuous cycle or the cycle will eventually be broken by a new AZDV routing packet update which corrects the loop.

In Chapter 8, loop detection and elimination methods were introduced with some success. However, while the measures eliminated the more severe routing loops, some loops remained. Moreover, the loop detection mechanism added a bandwidth overhead to each user message in the form of an attached list of previous forwarding nodes. Loop elimination was achieved by reverting back to SAGF behaviour upon detection of the loop. This was a simple and effective solution. The loops themselves are a symptom of an underlying shortfall in the AZDV system. It would be better to correct the shortfall, rather than correct the symptoms.

Node mobility was found to be a key cause of the routing loops. It created inconsistencies between the routing tables of adjacent nodes. It should be possible to eliminate the loops before they occur by detecting the underlying conditions which cause them (see Chapter 7). This can be performed by additional arbitration between adjacent nodes in the system. For instance, before forwarding a data packet, a node could broadcast the internal AZDV record pertaining to the destination of the data packet. Any potential forwarding nodes would then have the information needed to decide whether there is a discrepancy between the routing tables of the two nodes for this particular packet. Once the loop conditions are identified, pre-emptive action can be taken to prevent the loop before it occurs.

A possible future area of investigation for AZDV is scalability. AZDV has been demonstrated to work in a $1km^2$ grid section, but it will need to scale larger to work for cities such as Manhattan which is $59.47km^2$ (Wikipedia, 2009). Ultimately, for AZDV to be useful beyond simulation, it would need to be able to operate in large cities.

For scalability, Lee et al. (2008) suggests a technique of aggregating route-able junctions into larger route-able areas such as a rectangles. Route-able locations which are further away are routed to as an aggregated area. The routing table can contain one entry to address several routing zones. This technique can scale and is similar to an approach used by the GLS location service (Li et al., 2000), which uses quad-trees to organise groups of nodes. In the case of AZDV, an R-tree (Guttman, 1984) is a more appropriate data structure for representing a spatial hierarchy of routing zones.

To give a more concrete example, this technique could be used to group all route-able zones by

their respective city districts. The routing table for an AZDV node would feature an entry for each route-able zone in the local district and an entry for each city district. Therefore, to route to a zone outside the current district, it would simply have to lookup which district the zone inhabits and route to that district. The technique can scale up further i.e. cities each contain a set of districts.

9.3 Ethics

This section considers the ethical nature of AZDV, whether it is open to abuse or inherently enables unethical behaviour.

9.3.1 Privacy

Nodes in AZDV and SAGF use "hello" packets to identify local neighbours. This is the only instance in which nodes are individually identified in AZDV. Node specific information is not propagated in the protocol between nodes. The location of each car is only known by vehicles within wireless range. Even then, the location of each vehicle is only associated with the identity that the vehicle itself broadcasts, which could be changed at regular intervals should privacy be a major concern. AZDV does not compromise the privacy of the individual.

9.3.2 Security

As with any wireless network, in VANETs there is the opportunity for malicious or selfish behaviour. AZDV is a distributed routing system, which makes it vulnerable to false information. Depending on the security of the underlying VANET system upon which AZDV is built, there is the possibility of a Sybil attack (Douceur, 2002), (New, 2004), which could compromise the effectiveness of AZDV.

A Sybil attack is the act of impersonating or counterfeiting a node in the network, this would allow the attacker to plant false AZDV data in the system, misleading nodes about the best forwarding routes to a destination. An attacker could ensure that no messages ever reach their destination, or even redirect messages addressed to a certain location with the intent of intercepting them.

It is really down to the implementers and maintainers of the VANET to ensure the network is secure from attacks. It is down to those who use the network to ensure that they take the necessary precautions with any data they entrust to the network.

9.3.3 Safety

AZDV is a routing algorithm. It does not deal with any issues such as safety critical systems such as described in (Yin et al., 2004).

10 Appendix

10.1 Software Acknowledgements

The following tools were used in at least some small part in the process of creating this thesis.

- This entire thesis was compiled using LaTeX*www.latex-project.org/*
- All latex documents were edited in jEdit. *www.jedit.org*
- All C++ code was developed in Eclipse CDT. *eclipse.org/cdt/*
- C++ compilation was performed using gcc. *gcc.gnu.org*
- Bazaar was used for version control and backup purposes. *bazaar.canonical.com*
- All statistical analysis was performed using R. *www.r-project.org*
- Some diagrams were produced with Corel PhotoPaint 12
- Some diagrams were produced with Dia. *live.gnome.org/Dia*
- Cygwin *www.cygwin.com*
- CiteULike was used to manage the bibtex database for this thesis. *www.citeulike.org*

Bibliography

The Sybil attack in sensor networks: analysis & defenses, 2004. doi: 10.1109/IPSN.2004.1307346. URL http://dx.doi.org/10.1109/IPSN.2004.1307346.

An information propagation scheme for VANETs, 2005. doi: 10.1109/ITSC.2005.1520039. URL http://dx.doi.org/10.1109/ITSC.2005.1520039.

Reliable and Effcient Information Dissemination in Intermittently Connected Vehicular Adhoc Networks, 2007. doi: 10.1109/VETECS.2007.512. URL http://dx.doi.org/10.1109/VETECS.2007.512.

IEEE 802.11p: Towards an International Standard for Wireless Access in Vehicular Environments, May 2008. doi: 10.1109/VETECS.2008.458. URL http://dx.doi.org/10.1109/VETECS.2008.458.

Sumo Website. [ONLINE 10-2-09] http://sumo.sourceforge.net/, 2009.

M. Abolhasan. A review of routing protocols for mobile ad hoc networks. *Ad Hoc Networks*, 2(1):1–22, January 2004. ISSN 15708705. doi: 10.1016/S1570-8705(03)00043-X. URL http://dx.doi.org/10.1016/S1570-8705(03)00043-X.

Garcia Lunes Aceves. Loop-free routing using diffusing computations. *IEEE/ACM Trans. Netw.*, 1(1):130–141, February 1993. ISSN 1063-6692. doi: 10.1109/90.222913. URL http://dx.doi.org/10.1109/90.222913.

R. E. Bellman. *Dynamic programming*. Princeton University Press, 1957. URL http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/B0006AUXX8.

C. Bettstetter and C. Hartmann. Connectivity of wireless multihop networks in a shadow fading environment. *Wireless Networks*, 11(5):571–579, 2005.

E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1(1):269–271–271, December 1959. ISSN 0029-599X. doi: 10.1007/BF01386390. URL http://dx.doi.org/10.1007/BF01386390.

J. Douceur. The sybil attack, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.17.1073.

Kevin Fall. EECS 112, Lecture 17. URL: http://www.cs.berkeley.edu/~kfall/EE122/lec17/sld001.htm, 2009.

Antonin Guttman. R-trees: a dynamic index structure for spatial searching. *SIGMOD Rec.*, 14 (2):47–57, June 1984. ISSN 0163-5808. doi: 10.1145/602259.602266. URL http://dx.doi.org/10.1145/602259.602266.

A. Halati, H. Lieu, and S. Walker. CORSIM-corridor traffic simulation model. In *Proceedings of the Traffic Congestion and Traffic Safety in the 21st Century Conference*, pages 570–576, 1997.

X. Hong, K. Xu, M. Gerla, and Los C. A. Angeles. Scalable routing protocols for mobile ad hoc networks. *IEEE network*, 16(4):11–21, 2002.

E. P. Jones and P. A. Ward. Routing strategies for delay-tolerant networks. *Submitted to ACM Computer Communication Review (CCR)*, 2006.

Brad Karp and H. T. Kung. GPSR: Greedy Perimeter Stateless Routing for wireless networks. In *Mobile Computing and Networking*, pages 243–254, 2000. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.35.4735.

Michael Käsemann, Holger Füßler, Hannes Hartenstein, and Martin Mauve. A Reactive Location Service for Mobile Ad Hoc Networks, 2002. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.12.8778.

D. Krajzewicz, M. Hartinger, G. Hertkorn, P. Mieth, J. Ringel, C. Rssel, and P. Wagner. TheâĂİ simulation of urban mobilityâĂİ package: An open source traffic simulation. In *2003 European Simulation and Modelling Conference*, 2003.

D. Krajzewicz, M. Bonert, and P. Wagner. The open source traffic simulation package SUMO. *RoboCup 2006 Infrastructure Simulation Competition*, 2006a.

D. Krajzewicz, M. Bonert, and P. Wagner. The Open Source Traffic Simulation Package SUMO. *RoboCup 2006 Infrastructure Simulation Competition*, 2006b.

S. Krauss. Microscopic modeling of traffic flow: Investigation of collision free vehicle dynamics. *Deutsches Zentrum fuer Luft- und Raumfahrt. Forschungsberichte*.

S. Krauss, P. Wagner, and C. Gawron. Metastable states in a microscopic model of traffic flow. *Physical Review E*, 55(5):5597–5602, 1997.

P. Krishna, N. H. Vaidya, M. Chatterjee, and D. K. Pradhan. A cluster-based approach for routing in dynamic networks. *SIGCOMM Comput. Commun. Rev.*, 27(2):49–64, 1997. ISSN 0146-4833. doi: 10.1145/263876.263885. URL http://dx.doi.org/10.1145/263876.263885.

Kevin Lee. *Thesis Proposal: Advancing Geographic Routing in Vehicular Ad Hoc Urban Networks*. PhD thesis, 2009.

Kevin C. Lee, Michael Le, Jerome Harri, and Mario Gerla. LOUVRE: Landmark Overlays for Urban Vehicular Routing Environments. In *2008 IEEE 68th Vehicular Technology Conference*, pages 1–5. IEEE, September 2008. ISBN 978-1-4244-1721-6. doi: 10.1109/VETECF.2008.447. URL http://dx.doi.org/10.1109/VETECF.2008.447.

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, New York, NY, USA, 2000. ACM. ISBN 1-58113-197-6. doi: 10.1145/345910.345931. URL http://dx.doi.org/10.1145/345910.345931.

Nai W. Liu and Fang A. Liu. Search and Replication in a Hybrid Topology Architecture Based on RP(k). *Computer Science and Software Engineering, International Conference on*, 5:925–928, 2008. doi: 10.1109/CSSE.2008.66. URL http://dx.doi.org/10.1109/CSSE.2008.66.

C. Lochert, H. Hartenstein, J. Tian, H. Fussler, D. Hermann, and M. Mauve. A routing strategy for vehicular ad hoc networks in city environments. In *IEEE Intelligent Vehicles Symposium, 2003. Proceedings*, pages 156–161, 2003.

Christian Lochert, Martin Mauve, Holger Füssler, and Hannes Hartenstein. Geographic routing in city scenarios. *SIGMOBILE Mob. Comput. Commun. Rev.*, 9(1):69–72, January 2005. ISSN 1559-1662. doi: 10.1145/1055959.1055970. URL http://dx.doi.org/10.1145/1055959.1055970.

Tony K. Mak, Kenneth P. Laberteaux, and Raja Sengupta. A multi-channel VANET providing concurrent safety and commercial services. In *VANET '05: Proceedings of the 2nd ACM international workshop on Vehicular ad hoc networks*, pages 1–9, New York, NY, USA, 2005. ACM. ISBN 1-59593-141-4. doi: 10.1145/1080754.1080756. URL http://dx.doi.org/10.1145/1080754.1080756.

J. Mcquillan, G. Falk, I. Richer, B. Beranek, N. Inc, and M. A. Cambridge. A Review of the Development and Performance of the ARPANET Routing Algorithm. *Communications, IEEE Transactions on [legacy, pre-1988]*, 26(12):1802–1811, 1978.

Shree Murthy and J. Garcia-Luna-Aceves. An efficient routing protocol for wireless networks. *Mobile Networks and Applications*, 1(2):183–197, June 1996. ISSN 1383-469X. doi: 10.1007/BF01193336. URL http://dx.doi.org/10.1007/BF01193336.

Sze Y. Ni, Yu C. Tseng, Yuh S. Chen, and Jang P. Sheu. The broadcast storm problem in a mobile ad hoc network. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 151–162, New York, NY, USA, 1999. ACM. ISBN 1-58113-142-9. doi: 10.1145/313451.313525. URL http://dx.doi.org/10.1145/313451.313525.

Josiane Nzouonta, Neeraj Rajgure, Guiling Wang, and Cristian Borcea. VANET Routing on City Roads using Real-Time Vehicular Traffic Information. *IEEE Transactions on Vehicular Technology*, 58(7), 2009.

Charles E. Perkins and Pravin Bhagwat. Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers. *SIGCOMM Comput. Commun. Rev.*, 24(4):234–244, October 1994. ISSN 0146-4833. doi: 10.1145/190809.190336. URL http://dx.doi.org/10.1145/190809.190336.

Larry L. Peterson and Bruce S. Davie. *Computer Networks: A Systems Approach*. Morgan Kaufmann Publishers, second edition, 2000.

Antony Rowstron and Giovanni Pau. Efficient disruption tolerant discovery in dynamically partitioning wireless networks. 2009.

Charles L. Rutgers. An Introduction to IGRP. Technical report, The State University of New Jersey, Center for Computers and Information Services, Laboratory for Computer Science Research,, August 1991.

L. Smith, R. Beckman, D. Anson, K. Nagel, and M. Williams. TRANSIMS: Transportation analysis and simulation system. In *Conference: 5. National transportation planning methods applications conference, Seattle, WA (United States), 17-21 Apr 1995*, 1995.

Andrew S. Tanenbaum. *Computer Networks*, chapter 4. Prentice Hall PTR, fourth edition, 2003.

J. Tian, I. Stepanov, and K. Rothermel. Spatial aware geographic forwarding for mobile ad hoc networks. Technical report, University of Stuttgart, 2002.

A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks. Technical report, Duke University, 2000.

Wikipedia. Manhattan — wikipedia, the free encyclopedia, 2009. URL http://en.wikipedia.org/w/index.php?title=Manhattan&oldid=334872213.

B. Williams and T. Camp. Comparison of broadcasting techniques for mobile ad hoc networks. In *Proceedings of the 3rd ACM international symposium on Mobile ad hoc networking & computing*, pages 194–205. ACM New York, NY, USA, 2002.

Q. Yang. A Microscopic Traffic Simulator for evaluation of dynamic traffic management systems. *Transportation Research Part C: Emerging Technologies*, 4(3):113–129, June 1996. ISSN 0968090X. doi: 10.1016/S0968-090X(96)00006-X. URL http://dx.doi.org/10.1016/S0968-090X(96)00006-X.

Jijun Yin, Tamer ElBatt, Gavin Yeung, Bo Ryu, Stephen Habermas, Hariharan Krishnan, and Timothy Talty. Performance evaluation of safety applications over DSRC vehicular ad hoc networks. In *VANET '04: Proceedings of the 1st ACM international workshop on Vehicular ad hoc networks*, pages 1–9, New York, NY, USA, 2004. ACM. ISBN 1-58113-922-5. doi: 10.1145/1023875.1023877. URL http://dx.doi.org/10.1145/1023875.1023877.

J. Zhao and G. Cao. VADD: Vehicle-Assisted Data Delivery in Vehicular Ad Hoc Networks. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12. IEEE, April 2006. ISBN 1-4244-0221-2. doi: 10.1109/INFOCOM.2006.298. URL http://dx.doi.org/10.1109/INFOCOM.2006.298.