

REFINING AN ANT-BASED ROUTING PROTOCOL FOR
MOBILE AD-HOC NETWORKS

by

Nicholas Pilon

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
November 2007

© Copyright by Nicholas Pilon, 2007

DALHOUSIE UNIVERSITY

FACULTY OF COMPUTER SCIENCE

The undersigned hereby certify that they have read and recommend to the Faculty of Graduate Studies for acceptance a thesis entitled “REFINING AN ANT-BASED ROUTING PROTOCOL FOR MOBILE AD-HOC NETWORKS” by Nicholas Pilon in partial fulfillment of the requirements for the degree of Master of Computer Science.

Dated: November 29, 2007

Supervisor:

E. Grundke

Readers:

N. Zincir-Heywood

S. Sampalli

M. McAllister

DALHOUSIE UNIVERSITY

DATE: November 29, 2007

AUTHOR: Nicholas Pilon

TITLE: REFINING AN ANT-BASED ROUTING PROTOCOL FOR
MOBILE AD-HOC NETWORKS

DEPARTMENT OR SCHOOL: Faculty of Computer Science

DEGREE: M.C.Sc.

CONVOCATION: May

YEAR: 2008

Permission is herewith granted to Dalhousie University to circulate and to have copied for non-commercial purposes, at its discretion, the above title upon the request of individuals or institutions.

Signature of Author

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

The author attests that permission has been obtained for the use of any copyrighted material appearing in the thesis (other than brief excerpts requiring only proper acknowledgement in scholarly writing) and that all such use is clearly acknowledged.

Table of Contents

List of Tables	vii
List of Figures	viii
Abstract	ix
List of Abbreviations and Symbols Used	x
Acknowledgements	xii
Chapter 1 Introduction	1
1.1 Wireless Networks	1
1.2 Mobile Wireless Ad-hoc Networks	2
1.3 Snooping	3
1.4 Goals	3
Chapter 2 Literature Survey	5
2.1 802.11 Protocol Details	5
2.2 Standard Mobile Ad-Hoc Network Routing Protocols	6
2.3 Performance Reviews	8
2.4 Mobile Ad Hoc Network Theory	11
2.5 Snooping-Based Routing Protocols	12
2.6 Other Routing Protocols	12
2.7 Ant-Based Routing	13
2.8 Bee-Based Routing	15
Chapter 3 Ant Protocols	17
3.1 Basic Ant Protocol Operation	17
3.1.1 Sequence Numbers	20
3.1.2 Link Break Detection	20

3.2	Ant Protocol Parameters	21
3.3	Destroying Ants	22
3.4	Continuum Ant Creation	23
3.4.1	Local Link Break Rate Estimation	24
3.4.2	Continuum Ants Overhead Model	24
3.4.3	Feedback Loop	26
3.5	Lifetime Ant Creation	26
3.5.1	Link Break Event Refinement	27
3.6	Snooping Lifetime Ants	28
Chapter 4	Methodology	30
4.1	The Simulator	30
4.1.1	Why Not NS2?	32
4.1.2	Details of the Event Queue	33
4.1.3	Details of the Network Medium Layer	34
4.1.4	Details of the Network Link Layer	35
4.1.5	Details of Node Packet Processing	36
4.1.6	Details of the Network and Transport Layers	36
4.1.7	Details of the Application Layer	37
4.1.8	Details of the Routing Protocol Implementation	38
4.1.9	Statistics Gathering and Logging	39
4.1.10	Simulation Structure	39
4.2	Test Parameters	40
4.3	Recorded Results	42
Chapter 5	Results	44
5.1	Continuum Ants Algorithm Failure	44
5.2	AODV, LAnts, and SLAnts Test Results	44
5.3	Snooping Delay Analysis	51
5.4	Lifetime Estimation Accuracy	52

Chapter 6	Conclusions	54
6.1	Future Work	55
Bibliography	57
Appendix A	Other Figures	60

List of Tables

Table 4.1	Simulation Parameters	42
-----------	---------------------------------	----

List of Figures

Figure 3.1	Ant Protocol Operation	18
Figure 3.2	Ant History Operation	19
Figure 4.1	Simulator Architecture	32
Figure 5.1	Continuum VS. Lifetime Ant Populations	45
Figure 5.2	Packet Delivery VS. Number of Conversations	46
Figure 5.3	Percentage of Packets Delivered (5 conversations) VS. Movement Pause Time	47
Figure 5.4	Percentage of Packets Delivered (15 conversations) VS. Movement Pause Time	47
Figure 5.5	Percentage of Packets Delivered (25 conversations) VS. Movement Pause Time	48
Figure 5.6	Delivery Delay VS. Number of Conversations	49
Figure 5.7	Delivery Delay (25 conversations) VS. Pause Times	49
Figure 5.8	Packet Delivery Delay (with 25 conversations) VS. Conversation Length	50
Figure 5.9	Routing Packet Transmissions VS. Number of Two-Way Conversations	50
Figure 5.10	Queue Length Random Sample	52
Figure 5.11	Node Weights	53
Figure A.1	Delivery Delay VS. Conversation Interval	61
Figure A.2	Packet Delivery VS. Conversation Interval	61
Figure A.3	Packet Delivery VS. Conversation Length	62
Figure A.4	Routing Packet Transmissions VS. Conversation Interval	62
Figure A.5	Routing Packet Transmissions VS. Conversation Length	63
Figure A.6	Routing Packet Transmissions VS. Pause Time	63

Abstract

In this thesis, two new proactive routing protocols for use on wireless Mobile Ad-Hoc Networks are developed and tested using a discrete event-based network simulator. These protocols explore routing based on an ant based social insect metaphor. The use of snooping on wireless packet transmissions to augment routing protocol operation is also investigated. Prior work relied on global information to control the population of ants. The protocols presented in this thesis replace this global information with a completely local ant population control algorithm.

To test the performance of these protocols, a basic discrete event network simulator is implemented. Using this simulator, the proposed protocols are compared to AODV using several metrics in a variety of scenarios. The results indicate that the ant-based protocol is an overall improvement, with better performance and lower overhead.

List of Abbreviations and Symbols Used

ACK	Acknowledgement
AODV	Ad-Hoc On-Demand Distance Vector
CBR	Constant-Bit Rate
CoAnts	Continuum Ants Protocol
CTS	Clear to Send
DCF	Distributed Co-ordination Function
DSDV	Destination-Sequenced Distance Vector
DSR	Dynamic Source Routing
GPS	Global Positioning System
LAnts	Lifetime Ants Protocol
MANET	Mobile Ad-Hoc Network
MAR	Mobile Agent Routing
OLSR	Optimized Link State Routing
OSPF	Open Shortest Path First
PCF	Packet Co-ordination Function
QoS	Quality of Service
RREP	AODV Route Reply
RREQ	AODV Route Request

RRER	AODV Route Error
RTS	Request to Send
SLAnts	Snooping Lifetime Ants Protocol
TORA	Temporally-Ordered Routing Algorithm
ZRP	Zone Routing Protocol

Acknowledgements

I would like to thank Dr. Ernst Grundke for supervising my work on this thesis, and Dr. Nur Zincir-Heywood for stepping up to guide me through the final editing and defense process. I'd also like to thank my parents, my brother, and my friends for being helpful and supportive.

Chapter 1

Introduction

In this thesis, I develop two routing protocols for Mobile Wireless Ad-Hoc Networks. These protocols extend prior work on ants-based mobile wireless ad-hoc network routing protocols from the work of Zhou [31], Yue [30], and Thota [27] by altering the protocols to operate based only on information local to each node in the network. One protocol is the basic ants protocol with only the local information changes. The other further extends the ants protocol by using promiscuous-mode snooping to supplement routing traffic with information extracted from data traffic and routing packets sent to other nodes.

The goal of these new protocols is to use local information to regulate the population of ants and alter the size of that population in response to increases or decreases in node mobility and network load. I seek to show that the revised protocols using only node-local information still effectively respond to mobility-triggered network topology changes without drastic increases in routing protocol overhead. I also seek to show that promiscuous mode snooping on both routing and data packets can be a useful source of routing information.

1.1 Wireless Networks

Wireless networks have become widespread in recent years. Modern laptop computers almost always come with an integrated wireless networking card, and wireless routers and network cards for desktop PCs are cheap and reliable. The advantages of wireless networking are clear: users are not tethered to a wall-mounted network connection by a limited-length cable, but can move or place their computer anywhere within radio range of the access point.

The most common wireless protocol is 802.11. Several different varieties are available, with varying bandwidth and other features. The most common are 802.11b and

802.11g, which operate in the 2.4 GHz band and provide data rates of up to 11 Mbit/s and 54 Mbit/s, respectively. 802.11a operates in the 5 GHz band, with data rates of up to 54 Mbit/s and a slightly shorter effective range than 802.11b or 802.11g. The upcoming 802.11n standard operates in either band and provides rates of up to 248 Mbit/s at longer ranges than any of the prior standards. All of these standards follow the same basic protocol for unicast and broadcast transmission. The basic protocol is described in the IEEE 802.11 standard [4], and other standards documents extend this for the other varieties.

802.11 supports wireless networks regulated by a base station, or access point, and ad-hoc networks. When using an access point, network transmissions are regulated and routed by the access point. Access points can also serve as bridges to wired networks such as the Internet. However, access points are not always available in places where users may want to form a wireless network. 802.11's ad-hoc network support allows wireless devices to form a network and communicate without relying on a base station. In an ad-hoc network, all devices can participate in packet routing. This allows all nodes in the ad-hoc network to communicate with each other even when they are not in direct radio contact.

1.2 Mobile Wireless Ad-hoc Networks

Mobile devices are computers designed to be operable while the user is in motion, carried around with the user while active, or moved from place to place with the user as needed. They include laptops, palmtops, tablets, and cell phones. In many situations, including a conference, disaster area, or social space, users may want to communicate with each other using their mobile devices while going about their business, but may not have a wireless access point available to co-ordinate network activity and route packets. In these situations, the users' devices can form an ad-hoc network and communicate directly with each other. This is referred to as a mobile wireless ad-hoc network (MANET).

Every node participating in the MANET need not be within radio range of every other node. Nodes must be able to route packets for other nodes to get these packets to their destinations. Because the nodes can change position and network "links" are

based on radio range, a MANET routing protocol must expect the structure of the network to change, potentially frequently or rapidly, and adapt to these changes. A further constraint is that the routing protocol must not use too much bandwidth to adapt, or it risks crowding out user traffic.

1.3 Snooping

One of the protocols described in this thesis uses network packet snooping. Snooping refers to a device in a network that uses its network interface in promiscuous mode to listen to and process packets that are not addressed to it. In wired networks, snooping can only be performed on packets sent by other devices on the same segment as the snooper.

In a wireless network, every node within radio range of a transmitter can hear any packets that transmitter sends. In normal MANET operation, a node will put its wireless interface to sleep or discard any received transmissions not addressed to it when another node within radio range is communicating. When the interface is placed in promiscuous mode, it remains active and processes all received transmissions.

1.4 Goals

Prior simulations that were run to evaluate ant-based routing protocols in the work of Zhou [31], Yue [30], and Thota [27] relied heavily on global information to control the population of ants. Later work by Yue [30] and Thota [27] eliminated some, but not all, of this reliance on global information. The first objective of this thesis is to devise a mechanism for regulating the ant population that can operate based only on local information available to a single node in the network. The result is the Lifetime Ants (LAnts) protocol.

The second objective of this thesis is to extend the LAnts protocol by examining the methods for taking advantage of snooping in a routing protocol and the benefits and limitations of these methods. The revised protocol is called the Snooping Lifetime Ants (SLAnts) protocol. SLAnts attempts to improve the performance of the LAnts protocol by extracting useful routing information from the headers of packets it has

snooped on.

LAnts and SLAnts are compared against each other and the Ad-Hoc On-Demand Distance Vector (AODV) protocol [23]. AODV is a reactive routing protocol, and past studies have revealed it to be a consistently high-performance routing protocol for MANETs (See section 2.3).

Chapter 2 describes past work in MANET routing protocols. Chapter 3 introduces the new ant protocols and describes my goals for them in greater detail. Chapter 4 describes the tests performed to evaluate the routing protocols and the rationale behind them. Chapter 5 examines the results of the tests, and Chapter 6 details the conclusions I have drawn from these results and speculates about potential future work.

Chapter 2

Literature Survey

2.1 802.11 Protocol Details

The 802.11 protocol is detailed in IEEE Standard 802.11 [4], much of which is dedicated to addressing the problems of collision avoidance in the wireless medium. All nodes share a common transmission medium, and one node's transmission will block reception of any other transmission for all nodes within radio range of it. This leads to the "hidden node" problem, where the receiver of a transmission is within range of another node that is not within range of the original transmitter. In this case, the third node is not aware of the presence of the transmitter, and cannot hear any requests from the transmitter to reserve medium time.

One of the solutions to this problem used by 802.11's unicast packet transmission procedure is to use a series of packets transmitted from both the transmitter and receiver to reserve air time for the transmission around both nodes. This is called the Distributed Co-ordination Function (DCF). The transmitter begins the exchange by sending a Request To Send (RTS) packet to the receiver. If the receiver hears this packet, it listens for channel activity from other nodes. If it does not hear any activity in a specified window, it sends a Clear To Send (CTS) packet to the transmitter. This signals that it is safe to send a data packet. Both the RTS and the CTS packets include a field indicating how long the medium is to be reserved for. Other nodes that hear either packet will put their wireless interfaces to sleep until the transmission period has expired. Once the data packet has been received, the receiver sends an Acknowledgement (ACK) packet to the transmitter. The loss of any of these packets triggers a retry after a delay determined by a random exponential backoff algorithm. After a fixed number of retries, the transmitter assumes that the receiver is unreachable and discards the packet. This serves as a collision avoidance measure and solves the hidden node problem, because every node within radio range

of either participant in the transmission will hear either the RTS, the CTS, or both and avoid transmitting until the reserved period is over.

When transmitting a short packet after the medium has been idle for a sufficient interval, 802.11 allows a node to skip the RTS/CTS sequence and just transmit the packet and receive an ACK. 802.11 also defines a Packet Co-ordination Function (PCF) for collision avoidance, which requires the presence of an access point, and so cannot be used with ad-hoc networks, so it is not elaborated on here.

Broadcast packets do not use this sequence. Before sending a broadcast packet, a node watches for medium activity. If it does not hear any activity in a specified period, it sends the broadcast packet.

2.2 Standard Mobile Ad-Hoc Network Routing Protocols

Many different routing protocols have been developed for MANET routing. Here, I provide an overview of the operation of a selection of the standard, wide-spread protocols. These protocols are generally used as a basis for comparison for newly-developed routing protocols.

Destination-Sequenced Distance Vector routing (DSDV) [25] is a proactive hop-by-hop routing protocol that uses the classical distance vector method. Nodes periodically broadcast a packet to all of their immediate neighbours containing a list of all nodes they can reach and the costs of reaching those nodes. Sequence numbers are used to help prevent the formation of routing loops and to encourage the use of newer routing information. To further reduce the protocol's overhead, nodes can decide to broadcast "incremental" updates at more frequent intervals and complete updates at less frequent intervals.

The Optimized Link State Routing (OLSR) [6] protocol is a proactive hop-by-hop routing protocol for MANETs that uses the classical link state method. OLSR elects a group of "multipoint relays", such that every node is the neighbour of at least one multipoint relay and all multipoint relays are separated by at most one node. The multipoint relays are responsible for periodically flooding the network with link state information. This link state information is then used at each node to assemble an optimal routing table. In order to detect neighbours, OLSR uses a periodic "hello"

broadcast. This is a short packet that informs all other nodes within radio range of the node's existence.

The Zone Routing Protocol (ZRP) [13] is a hybrid proactive/reactive routing protocol. It divides the network into zones, and uses different routing protocols for routing traffic within zones and routing traffic between zones. Within zones, ZRP uses a proactive approach, providing a quick response to local topology changes and ensuring that nodes have a reliable picture of their immediate surroundings. Between zones, ZRP uses a reactive approach, which minimizes overhead.

The next two protocols both use cascading broadcasts. To perform a cascading broadcast, a node sends a broadcast packet to all of its neighbours. Each of these neighbours then repeats the broadcast, and each of their neighbours does the same until all nodes in the network have received the packet. Source-unique broadcast identifiers are used to ensure that a node does not re-broadcast the same packet twice.

Dynamic Source Routing (DSR) [15] is a reactive source routing protocol. Each DSR keeps a "route cache", which holds a complete route for each destination. When a packet is to be transmitted, the node looks up the packet's destination in its route cache and writes the route into the packet's headers. Each node along the route then examines this route to decide where to forward the packet next. When no route to a packet's destination is found in its cache, the node initiates route discovery using a cascading broadcast. Each packet in the broadcast keeps track of the nodes it has visited. When the desired destination receives the broadcast, it sends back a response packet with this route information to the source node. The protocol also takes steps to attempt to repair routes that have been disrupted by mobility, rather than requiring them to be rebuilt completely.

Ad-Hoc Distance Vector routing (AODV) [23] is a reactive hop-by-hop routing protocol. When a packet needs a route to a destination, it is placed in a queue and a Route Request (RREQ) packet is sent. Individual nodes keep track of what node they received the RREQ from, and use this to build a short-lived reverse route back to the originator of the RREQ. When the destination receives the RREQ, it sends a Route Reply (RREP) back to the originator along this reverse route. This RREP is used by

each node along the route to build proper entries in their routing tables. Sequence numbers are used to prioritize new information and avoid routing loops. AODV uses the same hello broadcast mechanism as OLSR to manage its list of neighbouring nodes.

When a link is broken, a Route Error (RERR) packet is used to inform all nodes using that route of the break. The nodes must then repeat the RREQ process to discover a new route to this destination.

2.3 Performance Reviews

Several groups have compared the performance of mobile ad-hoc network routing algorithms under a variety of circumstances using a variety of criteria to evaluate the performance of the algorithms. To avoid the complications of real networks and networking hardware, these tests were run in a simulated network. Using a simulator allows protocols to be examined and compared in a common “ideal” environment where network activity and simulation parameters can be precisely controlled and results can be directly monitored.

Broch et al. [5] examined the fundamental behaviour of four routing protocols: DSDV, the Temporally-Ordered Routing Algorithm (TORA) [22], DSR, and AODV. Their simulations examined the four protocols in a network of 50 wireless nodes in a rectangular space over 900 seconds with a “random waypoint” mobility model. In this mobility model, nodes move by choosing a destination in the simulation area and a movement speed from within an allowed range. Once a node reaches its destination, the process repeats. The node mobility rate, or how much disruption of network traffic node movement causes, is controlled by forcing nodes to pause for a fixed time at their destinations.

To simulate network traffic, Broch et al. used a Constant-Bit Rate (CBR) “application”. CBR sends packets of a fixed size to a destination at a fixed rate. Broch et al. used 10 to 30 CBR sources, all sending packets of a mere 64 bytes at a rate of four packets per second. They monitored the number of data packets delivered, the number of routing packets sent, and the deviation from the shortest path to a packet’s destination. They found that AODV and DSR had the best performance of

the four protocols, with the most packets delivered and the fewest routing packets sent, while DSDV and DSR created routes that were closer to the ideal route length.

The methodology of Broch et al. had serious weaknesses. They did not examine packet delivery delay, and the parameters used for their CBR application generated very small amounts of data traffic. Twenty nodes in their test network sent no traffic at all, which would favour AODV and other reactive routing protocols. These reactive protocols attempt to find a route only when a node first tries to send packets to another node or when a route is broken. A simulation with a small number of traffic flows with fixed participants is very advantageous for these reactive protocols. I examine this factor in my own simulations. See chapters 4 and 5 for details.

Johansson et al. [14] performed further work evaluating routing protocols in ad-hoc networks. This study looked more closely at the relative performance of reactive routing protocols and proactive routing protocols. Proactive routing protocols attempt to find routes between every pair of nodes in the network, even when there is no traffic exchange that requires it. Johansson et al. concluded that reactive protocols are superior, but all but one of their tests involved few CBR sources, leaving their methodology vulnerable to the same criticism as Broch et al. [5]. Their selection of routing algorithms included a single proactive algorithm (DSDV) and two very different reactive algorithms (DSR and AODV).

Larsson and Hedman, graduate students working under Johansson, explored and evaluated DSDV, AODV, and DSR in more detail [18]. They introduced a more complex measurement of node mobility and its impact on network structure: the mobility factor. The mobility factor measures mobility by examining the changes in distances between nodes. Like the other studies listed above, this study's simulations used a small number of traffic streams, favouring the reactive routing protocols.

The primary result of Larsson and Hedman's work was that link layer support is vital to good routing protocol behaviour. If the link layer does not provide notification of broken links, the performance of all protocols suffers, with greater delays and reduced packet delivery rates. Wired networks typically have the ability to detect a broken link at the physical layer. This is impractical in a wireless network, since a "link" is a side-effect of physical proximity between nodes. Instead, a link is treated

as having been broken when a packet transmission over that link has reached its retry limit.

Perkins et al. examined AODV and DSR with a more critical eye [24]. They identified several important problems with both protocols. They noted that in DSR a node that uses a stale route can pollute other nodes' route caches. They experimented with varying numbers of CBR traffic streams, and found that AODV fares poorly as the number of traffic streams increases.

Lee et al. examined DSR and compared it with the Distributed Bellman-Ford algorithm, also known as the Routing Internet Protocol [19]. Distributed Bellman-Ford did not respond to the MANET's mobility well, and showed significantly increased bandwidth use and computational overhead as the amount of mobility increased. The tests run with DSR showed a much more gradual decline as mobility increased.

Hsu et al. examined OLSR, Open Shortest Path First (OSPF), ZRP, AODV, and DSR [8]. They use four-hour simulations based on 19 mobile nodes and a single fixed node "base station". The nodes form two counter-rotating rings. Their results show that AODV is a slightly superior protocol in terms of packet delivery and throughput, but ZRP, OLSR, and DSR are close behind.

Ge et al. developed a method to add quality-of-service (QoS) routing to OLSR [9]. QoS routing requires that a routing algorithm not just provide a route to a destination, but provide a route that satisfies certain constraints or requirements. It might, for example, require a route that can sustain a certain amount of traffic per unit time for real-time voice or video communication. The QoS versions of OLSR developed by Ge et al. provide for this through two mechanisms. First, the multipoint relay selection takes into account node bandwidth using several different mechanisms. Secondly, the routing table uses available bandwidth on a route, rather than route length, when evaluating the quality of routes.

To test their algorithm, Ge et al. ran a simulation of their routing algorithm using 100 nodes in a 1000m by 1000m space. Rather than use a discrete event simulator, Ge et al. took a snapshot of the network and ran their QoS-modified OLSR algorithms, regular OLSR, and regular link-state routing on it, then evaluated the routes the algorithms chose. They found that two of their modified OLSR algorithms were

optimal. However, their limited simulation did not examine how their algorithms react to mobility or network congestion, though they do acknowledge this shortcoming and mention work to correct it.

Benzaid et al. integrated a mechanism for handling MANETs with fast-moving nodes into OLSR [3]. Their Fast-OLSR algorithm can coexist with OLSR in a single MANET, and changes only the neighbour discovery functionality. Their Fast-OLSR transmits a smaller “Fast-Hello” message with a shorter interval between transmissions than normal. This allows it to find new multipoint relays more quickly, and for its multipoint relays to detect that it has moved out of their range more quickly.

Benzaid et al. tested Fast-OLSR in a simulation with a small number of nodes and a single very fast-moving node. The fast-moving node’s speed ranged from 20 km/h to 150 km/h. They found that Fast-OLSR allowed for a packet loss rate of less than 10% at speeds of up to 90 km/h. No comparison was made to other routing protocols.

2.4 Mobile Ad Hoc Network Theory

Grundke and Zincir-Heywood produced two theoretical models for routing in ad-hoc networks. Their models approximate a MANET with a uniform continuum, ignoring variation in local situation between nodes. The first model places upper limits on the bandwidth available to individual nodes for data traffic and routing traffic and relates these limits to node density and transmission range. The two limits combined give an upper limit on all activity for a given node [12].

In the second model, Grundke relates the effect of node mobility and the arrival of routing information to the contents of the routing table [11]. The expanded theory predicts that the routing table in an ad-hoc network with a constant level of mobility will reach an equilibrium state, with a stable ratio of valid entries to invalid entries. This ratio is related to the level of mobility and the arrival rate of routing information. The proportion of valid entries in the routing table is given by

$$v = \frac{k_a}{k_a + k_m}, \quad (2.1)$$

where v is the proportion of valid entries, k_a is the arrival rate of routing information

per node, and k_m is the network-wide link event rate.

2.5 Snooping-Based Routing Protocols

DSR is the only one of the three common ad-hoc routing protocols (AODV, DSDV, and DSR) to make use of snooping [5, 14, 18]. As a source routing protocol, DSR requires that the complete route of a packet to its destination be determined before the packet is sent. This path is attached to the packet’s headers and is used by each intermediate node on the packet’s route to choose the next hop. Using snooping, other nodes can extract this path information and place it in its route cache.

Wu et al. developed a method for adding snooping to several other routing protocols as part of a set of modifications to optimize routes [29]. These optimizations include some snooping on extra information in data packet headers and were implemented under Linux for testing in an actual network. Their testing environment was very limited, using four laptops in pre-arranged configurations, with ftp and ping providing application traffic. They found that snooping provided an increase in performance, but the scope of their tests was so limited that no general conclusions can be drawn.

Jones et al. examined the power consumption of a variety of routing algorithms [16]. They report that “operating the receiver in promiscuous mode for caching and route response purposes resulted in high power consumption.” No analysis is presented of how much higher the power consumption becomes when performing snooping. Presumably, this increased power consumption is because the nodes cannot put their wireless interfaces to sleep while other nodes have reserved the medium for transmission.

2.6 Other Routing Protocols

Li et al. developed a routing protocol that uses geographic data [20]. Their protocol uses information from, for example, a Global Positioning System (GPS) service to build a map of the position of all nodes in a network. This map is then used to route packets so that they always move towards their destination. However, Ducourthial et

al. raise doubts about the performance of positional ad-hoc network routing schemes when faced with highly mobile nodes, particularly in a vehicular context [7].

Goff et al. introduced a routing protocol that uses signal strength to attempt to predict when a link will break [10]. They modified DSR with this prediction logic, and found that it significantly decreased the number of link failures by predicting them ahead of time and altering routes so that the links are no longer in active use when they fail.

Sinha et al. attempted to improve the performance of routing protocols by limiting active participation in the routing protocol to a handful of specifically-selected nodes [26]. In their routing core algorithm, a set of core nodes is elected by the network, and each core node is responsible for a set of subordinate nodes. Data packets are routed from the original transmitter to its associated core node, then that core node finds a path to the destination's associated core node. Finally, the destination's core node routes the packet to the final destination. Their experiments showed an improvement over the basic routing protocols they modified. The routing core algorithm achieves a 10% improvement in packet delivery percentage, and significant improvements in routing algorithm overhead in dynamic networks.

Sinha et al. also documented problems with broadcast transmissions. They examined cascading broadcasts or "broadcast storms", where the nodes that receive a broadcast packet re-broadcast it in order for it to reach every node in the network. They found that these broadcast storms caused widespread, significant interference with packet transmissions throughout the network, wasting bandwidth and requiring many packet retransmissions. Since 802.11 never attempts to retransmit a broadcast packet and broadcast packets do not use the RTS/CTS/ACK mechanism, they are very vulnerable to packet collisions. This was found to result in these supposedly-universal broadcasts missing some nodes in the network, particularly in high-mobility scenarios.

2.7 Ant-Based Routing

Minar et al. examined the use of mobile agents for network routing [21]. Their system involved "smart" agents carrying both program data and routing knowledge

exploring the network. Their analysis measured the level of connectivity achieved by their protocol in a wired network in a fixed configuration. They claim that their protocol achieves an average connectivity of 80%, but provide no indication of what this means or how it compares to other protocols. Their tests did not involve any other protocols, so there is no indication of whether or not this is significant.

Babaoglu et al. provided a detailed description of an ant-based routing algorithm for ad hoc networks called AntHocNet [2]. Like the agent-based system described by Minar et al. [21], their algorithm uses exploration to discover and maintain routes between nodes. Unlike the Minar protocol, AntHocNet’s ant packets contain only data, with no code component. AntHocNet is both a reactive and a proactive protocol. Like AODV, AntHocNet is reactive and does not attempt to obtain a route to another node until it needs to communicate with it. When it does, this route discovery is performed by means of a “reactive forward ant”, which is consecutively unicast or broadcast. When this reaches the destination of the route, a reply ant is sent back to the source, and sets up the route as it travels. AntHocNet adds a proactive component on top of this, with “proactive forward ants” that maintain active routes and explore alternate routes in case a link breaks. Their experiments showed that AntHocNet had superior delivery times, delivery ratios, and routing overhead to AODV and OLSR.

Zhou introduced a fully proactive ant-based routing protocol, the Mobile Agent Routing (MAR) protocol [31]. Despite its name, the MAR protocol’s ants are passive data packets, as in the work of Babaoglu et al. [2]. Instead of generating ants as needed to discover routes, MAR has a fixed population of ants constantly roaming the network and gathering routing information. MAR’s ant packets have a fixed-length history list, with each entry describing a node that forwarded the packet or an entry from the packet’s creator’s routing table. When a node receives an ant packet, it processes it, taking any information about better paths into its routing table and writing any information it has about better paths into the ant’s history list. This approach uses stigmergy, described by Babaoglu et al. [2]. Once the routing table and history list updates are done, the node unicasts the ant to a random neighbour. The population of ants is varied between scenarios, with more ants in scenarios with more mobility and fewer ants in scenarios with less mobility.

Neighbours are discovered by MAR using a periodic “hello broadcast”. Small packets are sent by nodes at a fixed interval, advertising its presence to other nodes within radio range. If a node does not hear a hello packet from a neighbour within a defined interval, it can assume that neighbour has somehow broken the link and removes it from its neighbour list.

Simulations showed that MAR offers an improvement over DSDV in terms of the percentage of packets delivered in some circumstances and a significant improvement in delivery delay in most circumstances. The paper did not measure routing overhead, which was likely high due to a fixed number of ants constantly traversing the network.

Yue further extended the MAR protocol by replacing the fixed ant population with fixed ant creation and destruction rates [30]. Yue compared this modified MAR algorithm with DSDV and AODV, and found that the modified protocol still offers improvements in delivery time and delivery delay over DSDV and AODV.

Thota provided a theoretical basis for the use of ants by using random walk mathematics to predict how far an ant will travel before it expires [27]. This was used to experiment with tuning some of the parameters of the MAR protocol from Yue’s work [30]. The continuum model from Grundke and Zincir-Heywood [12] provided some simple equations that could be used to adjust the behaviour of the MAR system to achieve a desired level of performance.

2.8 Bee-Based Routing

Wedde et al. took routing algorithms based on natural activity in another direction, creating an agent-based routing system based on the behaviour of bees [28]. Their algorithm borrows concepts from both DSR and AODV. Packets are kept in a queue on their originating node until a route is found to their destination. Conceptually, these packets are held by “packer” agents. “Scout” agents act like AODV RREQ/RREP packets. This uses the standard cascading broadcasts approach (see Section 2.2) to search for a destination, using an “expanding time to live timer” to control their propagation. Once the scout reaches its destination, it is unicast back to the original node.

When the scout packet returns to its original node, “Forager” agents are recruited

to carry packets along the route to the destination. Like DSR, Forager agents are given a complete route to their destination. Along the way, they monitor the nodes they pass through for either queue delay or remaining battery capacity. After delivering its packet, the forager agent waits at the destination until it can piggyback back to its source on other network traffic.

Wedde et al. ran several tests using a network simulator. The results of these tests showed that BeeAdHoc had similar throughput and packet delivery ratio to DSR, AODV, and DSDV, and was more energy-efficient. BeeAdHoc has also been implemented and tested on Linux [17]. The tests were run using both User-Mode Linux and a full kernel module, and both TCP and UDP. The tests showed that BeeAdHoc obtained performance equal to that of AODV with UML and UDP, but inferior with a full kernel module and UDP. With TCP, BeeAdHoc displayed superior performance under UML here as well. The improved power consumption of BeeAdHoc was observed throughout the tests.

Chapter 3

Ant Protocols

In this section, I present the details of two new ant-based routing protocols for MANETs: the Lifetime Ants Protocol (LAnts) and the Snooping Lifetime Ants Protocol (SLAnts). I also present a third protocol that proved impractical but was an important step in the development of LAnts, the Continuum Ants Protocol (CoAnts). These protocols extend the prior work done by Zhou [31], Yue [30], and Thota [27] on ant-based MANET routing. LAnts and CoAnts both feature minor improvements on basic protocol operation, but focus on providing a distributed method for managing the ant population based purely on node-local information. SLaNs takes this work and further extends it by modifying the routing protocol to gather extra information through promiscuous mode snooping.

3.1 Basic Ant Protocol Operation

All of the ants protocols described here employ the same basic behaviour. The initial behaviour described in this section was developed by Zhou [31], Yue [30], and Thota [27]. My extensions are described in subsequent sections. The goal of these ant-based protocols is to perform proactive route discovery in a MANET without using cascading broadcasts, which are both unreliable and wasteful [26].

All the ant-based routing algorithms gather and propagate routing information using ant packets. These packets keep a history list of nodes they've visited and the number of hops to reach these nodes from its most recent destination. These history lists are used to update a node's routing table. After updating its routing table from an ant history list, a node will update the hop counts for each entry in the history list, then add itself to the end of the ant's history list and randomly choose one of its neighbours to forward the ant to.

Nodes maintain their table of neighbours using periodic hello broadcasts to all of

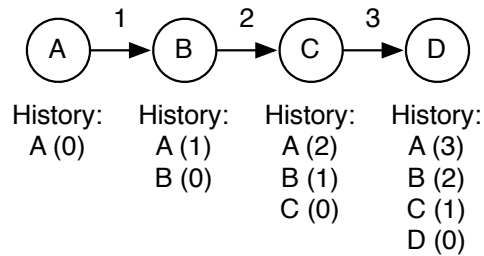


Figure 3.1: The basic operation of the ant protocol. The history list entries show the history list at each hop, while the numbered arrows show the hops the ant takes in sequence.

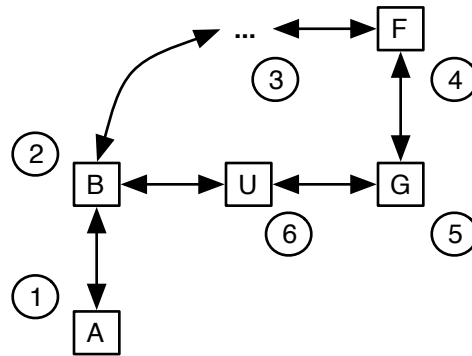
their immediate neighbours. When a hello packet is received, its source is entered into the table as a neighbouring node. When several hello packets transmission intervals go by without seeing a hello packet from a node, the node is assumed to have gone out of range and the link is marked as broken.

When a node updates its routing table from an ant, it examines each entry in the ant's history list. If the entry offers a shorter route to the destination than the route in the node's routing table, the ant's entry is used instead. Once this is done, the node updates the ant's history list based on its routing table. If it has a shorter route to a destination than the one reported in the ant's history list, the reported number of hops in the history list is updated to reflect the shorter route.

Figure 3.1 shows an ant building its history list as it moves from hop to hop. Both nodes add themselves to the end of the history list and hop counts are incremented at each step.

Figure 3.2 shows an example of how node routing tables are updated from ant history lists and vice-versa. Here, U has found a good route to A and B, but the ant has a shorter route to F than the one U is using. When the ant arrives, after having traversed the nodes in the numbered order, the ant brings news of the shorter path through G to F. U then updates its routing table from this information, using G as its next hop to F and marking the route as taking two hops.

The ant's history list is also updated. The ant took the long way around to U from A and B, and the information about the shorter path from the node's routing table is used to update the hop counts for these two nodes in the ant's history list.



Routing Table at U
at Ant Arrival:

Target	Next Hop	# Hops
F	B	5
G	G	1
B	B	1
A	B	2

Ant History List at
Ant Arrival at U:

Visited	Hops
G	0
F	1
...	...
B	5
A	6

Routing Table at U
after Ant Arrival:

Target	Next Hop	# Hops
F	G	2
G	G	1
B	B	1
A	B	2

Ant History List after
Ant Arrival at U:

Visited	Hops
G	1
F	2
...	...
B	1
A	2

Figure 3.2: The interaction between the ant history list and the node routing table. The ant, having moved through the displayed nodes in the numbered sequence, arrives at U and is processed there.

This gives an accurate picture of how many hops it takes to route packets through U to A and B, which may be useful wherever the ant goes next.

3.1.1 Sequence Numbers

The first improvement I made to the ants protocol, which is used in all of the protocols I present here, was to add sequence numbers. Sequence numbers are used in AODV and DSDV to encourage the use of new routing information and prevent the formation of routing loops [23, 25]. Each node stores the next sequence number for its routing information. When a node adds itself to an ant's history list, the node writes this stored sequence number as well and increases its stored sequence number by one.

Sequence numbers are used when updating routing tables and ant history lists during processing at a node. Information with a higher sequence number always takes precedence over information with a lower sequence number. When a link is marked as broken, the associated sequence number in the routing table is increased by one so that newer information must be used to restore the link.

The sequence number itself is an integer. The same method is used as AODV to prevent problems from occurring if the sequence number wraps around [23]. When incrementing the sequence number, the 32-bit value is treated as an unsigned integer. When comparing two sequence numbers, both are treated as signed 32-bit integers. The stored sequence number is subtracted from the sequence number for the incoming, potentially new, information. If the result is less than zero, the incoming information is treated as stale. Otherwise, the incoming information is treated as newer and preferred. This allows for a 32-bit sequence number to be used without overflow causing problems.

3.1.2 Link Break Detection

Notification from the link layer of link break events is vital to high-performance routing in MANETs [18]. Under 802.11, the link layer delivers a notification of a link down event when an attempt to transmit a packet exceeds the maximum number of retransmissions. The second addition I made to the basic ants routing protocol operation was to use notifications of link breaks to update a node's neighbour list and

routing table. Other than the source of the link break event, this link break event operates identically to the hello packet timeout link break event described in Section 3.1.

3.2 Ant Protocol Parameters

Though the basic operation of the ant-based routing protocol has been defined, my goal is to design a method to control the ant population based only on node-local information. Prior work used a global monitoring and regulating agent to create ants as needed to maintain a desired population [27, 30, 31]. To do this, I identify the parameters and values that affect the operation of the algorithm. The network structure places an upper limit on the ant history length, H , and defines the number of nodes, n , and the wait time at node, w . The routing protocol can adjust the per-node interval between ant creations, C , and the probability of ant destruction at a node p_t .

Each ant has a history list with a fixed maximum length, H . Since 802.11 does not permit link layer fragmentation of packets the MTU of the network is an effective upper bound on the size of each ant packet and, thus, of H . H can be smaller if desired (for example, a new ant in a small network), but cannot be larger.

n is the number of nodes in the network. Individual nodes cannot access this information directly, but can estimate it. From the perspective of a node, anything that is not in its routing table does not exist, since it cannot route or send packets to these unreachable nodes. Until they appear in its routing table, their influence over the node or its immediate neighbourhood is likely to be small. After a sufficient period of time, every node in the network should have a route (valid or invalid) to every other node. The number of entries in the routing table can be used as an estimate of n based only on node-local information.

w is the wait time at a node. Ants, like any other packet, take time to process. They also must sit in a queue waiting for any other packets ahead of them to be transmitted. Even if routing packets receive absolute priority, they still must wait for other routing packets at the node to be transmitted. Wait time should ideally be kept as low as possible, as the longer an ant sits in a queue, the more stale its routing

information gets.

The final parameters pertaining to the population of ants are the per-node interval between ant creations, C , and the ant destruction rate, which is defined probabilistically from a per-hop destruction probability, p_t . These are the parameters that control the ant population, and can be varied as needed by the routing algorithm. A higher birth rate or lower death rate will produce a larger ant population, while a lower birth rate or higher death rate will have the opposite effect. A larger ant population will deliver more routing information to each node more rapidly, compensating for the effects of mobility. A smaller ant population will deliver routing information less quickly, but will have a lower overhead.

As shown in Section 3.3, the rate of ant destruction can be effectively fixed. This means that the protocol can vary just C , simplifying the tuning logic and supporting mathematics.

3.3 Destroying Ants

The method for destroying ants is common to LAnts and CoAnts, and is based on the limits on the ant history length H described in Section 3.2. Once the ant has made H hops, the next hop will push the information from the first hop it made off the end of the history list, causing the ant to “forget” about it. At this point, the ant will forget one history list entry for each new one it discovers.

This implies that an ant’s effective lifetime is approximately, but not exactly, H hops. Using a probabilistic lifetime mechanism with an expected ant lifetime of H hops means that some ants will make more hops, carrying their routing information farther but “forgetting” some old information, and some will make fewer hops, consuming less network resources. If each node randomly decides not to forward an ant with probability p_t , an ant’s lifetime can be modelled as a geometric distribution,

$$\sum_{i=1}^{\infty} i p_t (1 - p_t)^{i-1}. \quad (3.1)$$

If the expected lifetime for an ant is H , then

$$p_t = \frac{1}{H} \quad (3.2)$$

is the per-hop ant destruction probability.

3.4 Continuum Ant Creation

The first algorithm, called CoAnts (Continuum Ants) I developed to try to determine an appropriate ant creation rate for network mobility is based on the continuum model developed by Grundke and Zincir-Heywood [11, 12]. In particular, the equation

$$v = \frac{k_a}{k_a + k_m} \quad (3.3)$$

predicts the fraction of a routing table filled with correct entries, v , given a routing information arrival rate, k_a , and a link break rate, k_m . Since k_a is determined by the ant population (and thus, as described in Section 3.2, the ant creation and destruction rates), if each node could estimate k_m (as in Section 3.4.1), this equation could be used to determine the ant creation rate needed to achieve a pre-determined target minimum level of routing table correctness, v .

Being able to do this is predicated upon being able to relate the ant creation rate to the routing information arrival rate. Let each node create ants using a random exponential distribution with a mean interval of C seconds/ant/node. Yue [30] gives

$$A = \frac{nT}{C} \quad (3.4)$$

as the expression for the total ant population, A , with n nodes and each ant having a lifetime of T seconds. Combining the per-hop wait time w and the per-hop termination probability p_t gives an expression for the expected ant lifetime in seconds,

$$T = \frac{w}{p_t}. \quad (3.5)$$

Combining the two equations gives a more useful expression for A ,

$$A = \frac{nw}{p_t C}. \quad (3.6)$$

Using this to find k_a requires examining the state at an individual node. The uniform continuum approximation used by Grundke and Zincir-Heywood [12] is useful for bridging the gap from the general case to a specific node. By ignoring edge conditions, all nodes can be assumed to face similar local network conditions. Because ants randomly move between nodes, each node receives an equal portion of the total

ants in the system per second. So the number of ants received by each node per second, a , with a per-node wait time of w is given by

$$a = \frac{A}{nw}. \quad (3.7)$$

Substituting in A gives

$$a = \frac{nw}{p_t C} \times \frac{1}{nw} = \frac{1}{p_t C}. \quad (3.8)$$

The arrival rate of routing information for each node, k_a , is given by

$$k_a = Ha \quad (3.9)$$

since each of the a ants arriving per second brings H pieces of routing information with it. Substituting in the prior expression for a gives

$$k_a = \frac{H}{p_t C}. \quad (3.10)$$

Substituting this into equation 3.3 gives

$$C = \frac{H(1-v)}{k_m v p_t}, \quad (3.11)$$

the final expression for C in terms of the target proportion of correct entries, v ; the estimated link break rate, k_m ; and the basic ant parameters identified in Section 3.2.

3.4.1 Local Link Break Rate Estimation

Using Equation 3.11 requires an estimate of k_m . Information about link events could be distributed directly, but this would require cascading broadcasts or other such mechanisms that the ants-based algorithms seek to avoid. Instead, a node can monitor its local link break event rate over a long period of time. By the assumptions of the continuum model, this can be generalized to the entire network by multiplying the locally-observed k_m , k_{local} , by the number of nodes in the network, n .

3.4.2 Continuum Ants Overhead Model

In order for a MANET routing protocol to be useful, the fraction of the network's resources used by the routing protocol must be kept as low as possible to prevent

routing traffic from crowding out user traffic. An expression for the total percentage of the channel capacity consumed by the transmission of routing information will allow the impact of the routing protocol on available capacity for user traffic to be measured. To begin with, assume that all nodes share the same channel, so a transmission by one node prevents all other nodes from transmitting until it is complete. This gives an approximate upper bound for the total fraction of the channel capacity consumed by routing traffic.

Each ant hop involves the transmission of a packet of HS bytes, where H is the history list size and s is the size of each history list entry in bytes. Suppose that the common channel shared by all nodes allows for the transmission of b bytes/second. From this, each ant transmission occupies the channel for $\frac{Hs}{b}$ seconds, plus a constant amount of packet overhead.

Since an ant is expected to make $\frac{1}{p_t}$ hops before it is terminated (Section 3.3), the total time an ant is expected to hold the channel over the ant's lifetime is

$$M = \frac{Hs}{p_t b}, \quad (3.12)$$

and the total time spent waiting at a node between transmissions is

$$W = \frac{w}{p_t}. \quad (3.13)$$

Taken over the ant's entire lifetime, the fraction of the channel's bandwidth occupied by one ant is

$$B_{one} = \frac{M}{M + W}. \quad (3.14)$$

This can be generalized to apply to A ants:

$$B_A = \frac{MA}{MA + W + (A - 1)\frac{Hs}{b}}. \quad (3.15)$$

This assumes that the collision avoidance algorithm in use manages to arrange for a perfect round-robin transmission scheme. It is possible for collisions to make this worse.

Expanding the terms of this expression obtains

$$B_A = \frac{Hsnw}{Hsnw + wp_t bC + (A - 1)Hsp_t^2 C}. \quad (3.16)$$

This shows that increasing the per-node ant creation interval, C or decreasing the per-hop probability of ant destruction, p_t will cause an increase in routing protocol overhead.

3.4.3 Feedback Loop

The CoAnts (Continuum Ants) algorithm has a significant flaw: Equation 3.6 features a feedback loop. Once the number of ants, A , increases above the number of nodes in the network, n , every node in the network has at least one ant in its queue, and some have more than one ant. Since w , the wait time, includes queue delay, this means that w increases as A increases. But, by Equation 3.6, an increase in w requires an increase in A . The full implications of this are demonstrated in Section 5.1, but they require another algorithm to be developed for managing the ant population.

3.5 Lifetime Ant Creation

The Lifetime Ants algorithm is directly based off the observation from the failure of the CoAnts algorithm: it is not useful for the total ant population, A , to be bigger than the number of nodes in the network, n . Once $A > n$, there is always at least one queue with two ants in it. The information carried by the second ant is getting stale, and the second ant is doing little to gather new routing information but is consuming just as much in the way of network resources as the first ant.

To correct this problem, I developed the concept of ant “ownership”. Each ant is “owned” by the node that created it. This concept does not imply any actual control or software or hardware connection, and is purely theoretical. If every node created a new ant as soon as the ant it owns is destroyed, the number of ants in the network would be constant, $A = n$. This places a strict limit on the number of ants in the network. Instead of adjusting the ant creation rate to compensate for mobility, the ant creation rate is adjusted based on network load.

Even $A = n$ might still be more ants than needed. Prior ant simulations by Zhou [31] involved 20 ants in a 50 node scenario and obtained similar packet delivery levels to other MANET routing protocols. To provide more fine-grained control, I added X , the ant population scaling constant. Instead of having $A = n$, the algorithm has

$A = nX$. With a fractional X , this reduces the number of ants below n .

In an actual network the concept of ownership cannot be used to control the ant population. Making information about the status of its ant available to a node would consume network resources. It is possible to produce the same behaviour through an estimate of the ant’s “wall clock” lifetime, l , measured in seconds. l can be calculated from ants as they are processed by adding an extra field to the ants’ history entries. Using this system, the mean ant creation interval is determined by

$$C = \frac{l}{X}. \quad (3.17)$$

The actual creation interval is randomly varied using the same exponential random distribution as CoAnts.

In order for this algorithm to work, nodes must be able to obtain a good estimate of l . This is derived by using a weighted exponential average of estimates obtained from individual ants. Each ant has two extra fields: h , the number of hops it has made so far, and t the total time it has spent in queues and being transmitted. From this, the expected time per hop can be obtained using

$$l_{hop} = \frac{t}{h}. \quad (3.18)$$

Combining this with p_t gives an expression for the expected lifetime of the ant in seconds:

$$l_{ant} = l_{hop} \times \frac{1}{p_t}. \quad (3.19)$$

LAnts primarily attempts to adjust the performance of the routing algorithm to compensate for network congestion and load. As delays grow, so does the interval between ant creation events, and the amount of bandwidth consumed by the routing protocol decreases. When delays decrease again, the interval between ant creation events shrinks with them. LAnts does not attempt to compensate for mobility, and instead allows routing performance to decline as mobility increases.

3.5.1 Link Break Event Refinement

One interesting discovery of my preliminary test simulations with LAnts was that “link break events” are expensive. The most direct and immediate indication of a

broken link is a failed packet transmission. Because the link layer must employ a retry-and-backoff algorithm to prevent failures due to collisions, broken links take some time, t_{lb} , to detect [4]. This imposes a minimum resolution on routing information, since link breaks cannot usually be detected any faster than this. This means that originating ants more frequently than t_{lb} seconds is counterproductive. This imposes an effective lower bound on the ant creation interval.

3.6 Snooping Lifetime Ants

The Snooping Lifetime Ants (SLAnts) protocol extends the Lifetime Ants (LAnts) protocol by modifying the protocol to gather additional routing information through snooping. The packets that can be captured through snooping fall into two general categories: ants sent by the routing protocol and data packets sent by applications.

A node can snoop on data packets and examine their headers to extract useful routing information. From a packet's IP header, a node can discover the following information:

1. The presence of the node transmitting the packet. This can serve as a replacement for a hello message, meaning that nodes actively engaged in packet transmission can increase the interval between their hello messages.
2. The presence of the packet's intended receiver. This node may be a neighbour of the snooping node, or it may be reachable in two hops through the transmitting node. This node might have since moved, breaking this link, so this is more tenuous than the first piece of information.
3. The presence of the source node of the packet. The source node must be reachable through the transmitter, or have been reachable through it lately, since this packet has somehow made its way from it to the transmitter. If all nodes use the same initial hop limit for IP packets, the number of hops can be determined from the packet's hop count field.

None of the information gained from snooping on IP packets has a sequence number attached, so the Node Ownership algorithm always prioritizes information from routing packets over it.

Unlike IP packets, a node does not have to perform any extrapolation on snooped ant packets. These packets already contain routing information, which can be processed and integrated into the routing table just like any other ant packet. The only difference between the two procedures is that ants that are overheard using snooping are not forwarded.

Chapter 4

Methodology

4.1 The Simulator

I evaluated the performance of the ant-based routing algorithms using a network simulator. The network simulator is a discrete event simulator that I wrote in Python. An open-source network simulator called NS2 [1] with similar functionality was available, but prior experience working with it was extremely negative. See Section 4.1.1 for a more detailed explanation of the issues encountered and the reasons I had for writing my own simulator. My goal was to create a simulator that I could use to run the necessary scenarios for this thesis. My secondary goals were to create a simulator that was more modern, cleaner, and easier to extend than NS2. I chose to implement the simulator in Python because in my prior experience with the language, I had found Python to be extraordinarily expressive and powerful without sacrificing much in the way of speed or efficiency.

The simulator's overall architecture is shown in Figure 4.1 and described here in general terms. Subsequent sections provide further details for each specific component. The simulator uses a single central event queue, which dispatches events sequentially. This provides an ordering for events, and allows for events to be scheduled to occur at some future time.

The operation of the network simulator itself revolves around two classes: the network medium and the network node. A network medium enables communication between two or more nodes. In a wired network, a network medium would be a single cable, or multiple cables joined by a bridge, repeater, hub or other low-level networking hardware. In a wireless network, there is a single medium, which keeps track of the position and radio ranges of nodes using it to communicate. In the TCP/IP model, this is Layer 1, the physical layer.

The network node represents a single device in the network, and joins together all

the protocols, applications, and links that are “running on” or otherwise “part of” that node. The node is outside of the layer model, but joins together layers 2 through 5 of the TCP/IP model, providing common services for the simulations of all of these layers.

Links receive packets from the medium and pass them to the node for processing, and are passed packets by the node to send using the medium. The links simulate layer 2 in the TCP/IP model, the data link layer. A node can have one or more links, but a wireless node will typically only have one, its wireless interface.

Protocols decide what to do with traffic packets that have been received by the node. Common choices include passing the packet to a higher layer for processing or forwarding or discarding the packet. Protocol layers are not linked directly to each other. Instead, the node relays packets between protocols as needed. When a protocol has decided to pass an inbound packet to a higher layer, it is sent back to the node, which, in turn, finds the appropriate protocol to continue processing the packet based on the packet’s type.

For example, a UDP packet contained in an IP packet will first be processed by the IP protocol. When the IP protocol decides that the packet is to be further processed by this node, it will send the data packet carried by the IP packet back to the node. In this case, this is a UDP packet, which the node will relay to the UDP protocol.

Similarly, packets being passed down the stack for transmission are dispatched to their next layer for processing by the node based on the type of address associated with the transmission. Protocols cover TCP/IP model layers 3 and 4, the network and transport layers, and also include routing protocols, address resolution protocols, and anything else that works with “raw” packets or manipulates packets directly.

The next piece of the network simulator is applications, simulating layer 5 of the TCP/IP model. Applications both originate packets and serve as the final destination for packets, and are handled separately from protocols for simplicity. Unlike protocols, packets are never automatically relayed to an application by the node. A protocol must explicitly deliver a packet to an application. This provides a similar behaviour to sockets and other application-level networking APIs.

The correctness of the simulator was verified by repeating some of the experiments

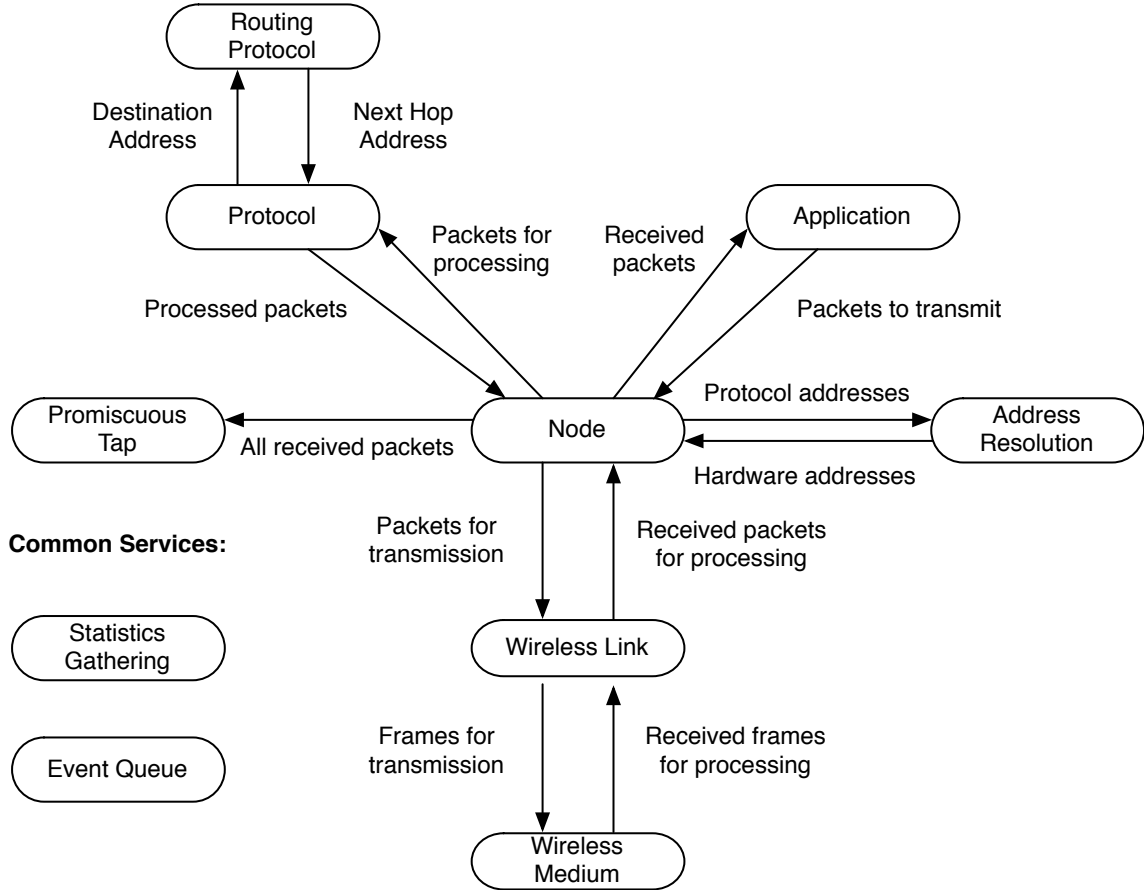


Figure 4.1: The general architecture of the simulator used to test LAnts and SLAnts. Lines represent the major packet flow between components.

used to test the older global knowledge-based ants routing protocols using the new simulators. The results of these simulations were compared to the results previously obtained by Zhou and Yue [30, 31], and found to be similar. This shows that the simulator produces similar results to, and is no less valid than, NS2 for the scenarios examined for this thesis. Since NS2 is widely used in other MANET routing studies [5, 14, 18], it was judged to be a suitable basis for comparison.

4.1.1 Why Not NS2?

I chose not to use NS2 for this work because experience with it in prior thesis work was not positive [27, 30, 31]. During this prior work, other students found that the results

of simulations tended to change significantly with new NS2 releases, documentation was inadequate, and the logging and statistics format was obtuse. The changing simulation results were particularly troublesome, and made tracing the causes of problems unnecessarily difficult. The sheer age of NS2’s code meant that new gcc or tcl releases tended to prevent it from compiling, necessitating an upgrade to a newer version. In turn, this newer version would behave completely differently, and the changelogs and other documentation would provide no indication of what had changed.

The process of implementing a routing protocol for simulation using NS2 was also largely undocumented. The interface the routing protocol object needed to implement had to be reconstructed by reading the source for other routing protocols. Even once this implementation work was done, adding the protocol still required making undocumented edits to several “internal” data structures, and which data structures needed to be edited changed from release to release.

Writing a new simulator gave me better knowledge of all layers of the simulated network. It helped me to understand how wireless physical and link layers operate. This proved invaluable when evaluating the failure of the continuum model-based ant population algorithm (see Section 5.1) and the role of link layer timeouts in ant creation (see Section 3.5.1).

4.1.2 Details of the Event Queue

The event queue is a simple priority queue of event objects. Each upcoming event has an associated time, which is used to order the priority queue. The simulation moves through this queue, popping events and executing them in order. Each event object contains a reference to a bound method and a map that links arguments to that method to their values. No other limitations are imposed on the method. It can do anything, including schedule other events as desired. A global clock, maintained by the simulator, is used to enforce the flow of time by preventing events from being registered before the “current” time. Once an event is executed at a given time, no more events can be added to the queue for execution before that time.

The event queue itself has no inherent time granularity, it simply requires that

all objects submitting events to it use the same time scale. The rest of the simulator uses a millisecond time scale, but individual times can be as precise (or imprecise) as allowed by Python’s floating point data type.

A common pattern is a method that reschedules itself at a later date. This is used to create a “timer” which allows an activity to be performed repeatedly.

4.1.3 Details of the Network Medium Layer

The network medium layer is responsible for routing transmissions between nodes. Transmissions are passed to the network medium from network links, and back to nodes through network links. All of the tests run for this thesis used a wireless network medium. The wireless medium tracks the position of nodes in it, and monitors the transmission status of each node: clear, busy, or interference. When a transmission begins or concludes, the status of all nodes within range is checked, and changed to indicate a collision if one has occurred.

The range of a transmission is determined by a plug-in module. For the tests run for this thesis, I used a module based directly on the model and logic used by NS2. It determines the power of the received transmission based on the original transmission power and the distance between the two. The Friis free-space equation is used for transmissions at ranges shorter than a computed cross-over distance, while the two-ray ground reflection model equation is used at ranges beyond this distance. Propagation delay is ignored, since it is insignificant at the distances in the scenarios I ran.

The Friis free-space equation,

$$\frac{P_{recv}}{P_{trans}} = G_t G_r \left(\frac{\lambda}{4\pi RL} \right)^2 \quad (4.1)$$

describes the power of the signal at the transmitting and receiving antennas, P_{trans} and P_{recv} , in terms of the gain for the transmitting and receiving antennas, G_t and G_r ; λ , the wavelength of the transmission; L , the system loss; and R , the distance between the two antenna.

The two-ray ground reflection model equation,

$$\frac{P_{recv}}{P_{trans}} = \frac{G_t G_r h_t^2 h_r^2}{d^4 L} \quad (4.2)$$

is more accurate at longer ranges, and models both the direct path and the ground reflection path. This introduces two additional values, h_t and h_r , the height of the transmitter and receiver, respectively.

As in NS2, G_t , G_r , and L are all set to 1 in my simulator.

The wireless medium also supports several kinds of polling, allowing a network link to check its status and monitor the medium for nearby traffic.

4.1.4 Details of the Network Link Layer

The network link layer is responsible for mediating between the network medium layer and the node objects. All of the tests run for this thesis used an 802.11-derived link layer. This link layer implements only the basic congestion control mechanism, without any of the more complicated network management aspects of the protocol. The simulations I ran were concerned exclusively with basic network operation, and higher-level network management activity was outside the scope of my experiments.

Packets waiting to be transmitted are kept in a queue. The simulator does not limit the length of this queue. When a packet transmission attempt exceeds its retry limit, the destination node is considered to be unreachable. To avoid wasting time and network resources on repeated transmission attempts to reach this node, all packets for that destination are removed from the queue.

Normally, links only pass incoming packets that are addressed to them up to their nodes for processing. When one or more promiscuous taps are present on the node, the link puts itself into promiscuous mode and passes all received packets to the node for processing. The node, in turn, filters these packets so that promiscuous taps receive all of them, while protocols only receive ones addressed to one of this node's links.

Links also provide reports on the success or failure of packet transmission to higher layers. This information is ignored by default, but can be used by wireless network protocols to discover information about the status of a node's neighbours.

4.1.5 Details of Node Packet Processing

The node object handles all the necessary work to simulate a device that falls outside of the standard TCP/IP layer model. It joins together all the applications, protocols, and links running on a single node, providing look-up and relaying services. This means that each protocol, application, and link only has to talk to a single node object, rather than having to manage a collection of objects from the layers above and below it.

The node's primary responsibility is passing packets between protocols. When a node is handed a received packet, it searches through its protocols until it finds one willing to accept responsibility for processing the packet. When a node is handed a packet for transmission to a specified destination address, it fills the same role, but this time searches for a protocol that can handle transmission to that kind of destination address.

In addition to these responsibilities, the node provides address resolution services, so that network layer addresses used by protocols can be translated into hardware layer addresses used by links. The actual Address Resolution work is performed by a "plug-in" module, so that different address resolution protocols can be used as needed.

4.1.6 Details of the Network and Transport Layers

The first protocol encountered by most packets during processing at a node is the network layer. I used a simple implementation of IPv6 for this thesis. IPv6 is the upcoming replacement for IPv4, and was chosen for its simplified specification and ability to handle extension headers. These extension headers could prove invaluable in future work, as described in Section 6.1.

The implementation provides only basic routing and delivery functionality, as the more complex control functionality is unnecessary for the scenarios used. Nodes in these scenarios never attempt to ping each other, trace packet routes, or do anything beyond transmit and route packets.

In order to route packets, the network layer protocol uses a routing protocol object. When a packet is to be routed, the packet and any other relevant information are

passed to the routing protocol, which returns the protocol address of the next hop and the link to use to send the packet to that hop. Gathering the information needed to provide this information is the responsibility of the routing protocol object.

After the network layer, most packets are passed up to the transport layer. The functionality here is very similar to the network layer functionality; the packet's transport layer headers are removed and the packet is dispatched appropriately to additional protocols or applications. I chose to use UDP for the transport layer for the simulations for this thesis. The UDP transport layer provides methods to allow applications to bind themselves to specific ports, and dispatches packets to applications based on these bindings.

4.1.7 Details of the Application Layer

Applications are registered with nodes separately from protocols. Protocols must dispatch packets to them explicitly. The only application used in this thesis is the CBR application, which sends data at a constant rate. I implemented several variations of it, each using different schemes for switching the destination of the CBR stream and managing the CBR packet rate. The first application implemented sent packets at a constant rate to a single destination for the entire duration of the simulation. The second application implemented worked its way through a circular queue of destinations, sending packets to each in turn at a constant rate for a pre-defined duration.

The implementation used for the main test simulations revolves around a queue of packet flows. The packet flow queue is populated before the simulation starts. Each packet flow has a destination, start time, and duration. These control when the application starts the CBR flow to this destination and how long it lasts. This application was used to implement the “conversation” model used for the simulations, with two nodes randomly chosen at the start of the simulation exchanging packets for a specific duration, then pausing before choosing new partners and continuing their data transmission.

CBR packets have no actual content. They are merely objects that claim to have a specific size when queried. This size, plus the size of any envelope packets added

by other layers, is used to determine how long a packet will take to transmit.

4.1.8 Details of the Routing Protocol Implementation

The implementations of LAnts and SLAnts serve as both Address Resolution objects and Routing Protocol objects. Timers are used to handle the regular transmission of both hello messages and newly-created ants. The time of the next ant creation event is computed when an ant is sent. If the protocol's estimation of the proper period between ant creation events changes between the two events (see Section 3.5), this is not taken into account until after the next ant is created. The estimations are recomputed whenever new information is available.

Since Hello packets already contain information about the hardware and protocol addresses of the nodes they come from, both Ants protocols can be used in place of a standard address resolution protocol like ARP. The protocol object's Address Resolution records are maintained using information from hello packets. These packets inform a node of the existence of another node, and contain the necessary information about hardware and protocol addresses. This information is also used to maintain a list of neighbours to whom packets and ants can be forwarded. When a node has no neighbours, ant creation is suspended until a new neighbour is discovered.

To receive packets, the LAnts implementation registers itself as a protocol. Since ant packets contain all the information needed for transmission and only make one hop at a time, the ants are transmitted from node to node without using IP or any other network-layer protocol. This means that all nodes in the network must implement the ants protocol, or they will not know how to forward these packets, creating gaps in the network.

SLAnts performs its snooping duties by also registering itself as a promiscuous tap. Both protocols obtain the necessary information about dropped links by requesting notification of failed packet deliveries from the link layer. When the protocol object is informed that a packet delivery has failed, the corresponding neighbour is marked as no longer available.

Routes and address resolution information have an expiry time associated with them. Expiry is handled lazily. Before any routes are accessed or addresses are

resolved, the tables are scanned for expired entries and any found are purged.

In order to perform its lifetime estimation, LAnts requires that each ant carry information about how long it spent being processed and waiting in the queue at each node. This information can only be written into the ant when it is passed from the link layer to the physical layer for transmission. A callback registration system is used to implement this, with the link layer triggering the callback the first time it tries to send the packet. This means that the estimates are not perfectly accurate, since they do not include the additional time spent waiting if the first transmission fails.

4.1.9 Statistics Gathering and Logging

The results of the simulations are monitored and recorded using statistics gathering and logging mechanisms. Statistics gathering is the main mechanism of monitoring simulation results. When a particular situation of interest occurs, such as a packet delivery, a call is made to the statistics module to record it. Several kinds of statistics are available, but the most common one tracks the time and source node of the event. This allows statistics files to be analysed based on time or source, or both together.

Originally, I was going to use Python's logging module to provide a detailed trace of simulator activity, independent of any particular statistics being gathered. Preliminary tests revealed that Python's logging module was far too heavy-weight to be used for this. Even when logging was turned off entirely, the logging calls more than doubled the run time of a simulation. Logging is only used to monitor for inconsistent simulator behaviour, and all logging calls except for those providing notification of errors were commented out during full simulation runs.

4.1.10 Simulation Structure

This network simulator does not use a separate simulator description or configuration file. A simulation is written as a Python program, which creates, configures, and connects objects as appropriate to construct the simulated nodes, assemble a scenario, and record statistics. Movement scenario information is the only parameter read from a configuration file. This is standard practice for Python programs. Python's syntax

is relatively simple and easy to read, and anyone doing significant work with the simulator must learn it anyway. Using a custom format for scenario files would have increased overhead significantly, due to the need to design the format, then design and write software to process that format. Python also has the advantage of flexibility and power, with conditionals, looping constructs, and powerful list-processing ability built-in.

Even NS2 recognizes these advantages. It uses TCL programs to assemble scenarios rather than a custom format, for similar reasons.

I wrote a simple scenario generator to quickly create the large number of scenarios needed for my tests. (See Section 4.2 for information on the tests run.) This scenario generator used simple template scenario code and filled in the parameter values based on command-line arguments for each test.

Movement scenario generation proved exceptionally difficult. Nodes must move in such a way that the network does not fragment at any point. In order to achieve this, I used the waypoint mobility generator from NS2 to build the movement scenarios, then wrote my own code to interpret and utilize this movement information.

4.2 Test Parameters

To examine the behaviour of the routing protocols, I built 180 simulations to compare the protocols under different conditions. These conditions are summarized in Table 4.1. The simulations were constructed using five variable parameters, with two to five values each. All combinations of parameter values were tested. For each simulation I ran three tests, one with each routing protocol I wanted to examine: LAnts, SLAnts, and AODV. The number of nodes was fixed at 50 and the length of the simulation at 900 seconds, and the other parameters were chosen to control node network activity and mobility. These two basic parameters were chosen based on the work of Broch et al. [5]. A 900 second simulation minimizes the influence of the “start-up time”, when all nodes are switched on at the same instant. A network size of 50 nodes allows for some complexity in terms of routing and mobility without having an unmanageable number of participants.

Mobility was primarily controlled through pause time. The mobility model is a

waypoint mobility model, similar to the one used by Broch et al. [5]. Nodes move between waypoints with a speed randomly chosen from a small range. When they arrive at a waypoint, they pause for a fixed amount of time before moving to their next waypoint. Five different random waypoint movement patterns were generated for each pause time, making for fifteen total movement patterns. Pause times of 0, 60 and 300 seconds were used. With a pause time of 0 seconds, the nodes in the network are constantly in motion. A pause time of 300 seconds means they almost never move. A pause time of 60 seconds gives behaviour between the two. The maximum movement speed of the nodes was set to 5 m/s, just over twice human walking speed. The movement scenario generator randomly chose a speed for movement between successive pairs of waypoints.

Nodes were distributed in a 1400m by 300m area. Using a rectangular area ensures a less regular distribution of nodes, and encourages the formation of longer paths [5]. The NS2 waypoint mobility generator (see Section 4.1.10) distributes nodes and waypoints randomly, but ensures that the network remains a single component throughout the entire simulation. The transmission range was set at 250m because the NS2 waypoint mobility generator is hard-coded to use this range.

Application traffic packets were controlled through three parameters. The most significant was the number of “conversations”, with each conversation being a two-way CBR traffic flow between two randomly-chosen nodes. The tests involved 5, 15, and 25 conversations. While end-points were randomly chosen, more conversations mean that more nodes in the network are likely to be involved in sending or receiving traffic. The length of the conversations was also varied, with lengths of 100 and 200 seconds. I also used a break between conversations of either 30 or 60 seconds. The intent of these two parameters was to provide more fine control over the data traffic. Longer conversations would give routing protocols more time to adjust to traffic, and longer break times would allow nodes to empty out their packet queues.

Packet size was fixed at 512 bytes, with 4 packets sent per second by each participant in a conversation. This was chosen to attempt to stress the network more than the 64 byte packets sent at the same rate in prior work. Data packets were sent using UDP. UDP was chosen over TCP because TCP adjusts packet rate based on

network congestion and other sources of packet loss. This means that it can adapt to network congestion problems created by poor route selection or high routing protocol overhead.

Combining all values for all parameters gives 180 simulations for each routing protocol: five movement patterns, three pause times, three conversation counts, two conversation lengths, and two conversations intervals.

Combined, these factors allowed me to vary both network load and several aspects of the nature of the conversations. I expected simulations with few conversations or long-lasting conversations to favour AODV, since its high overhead when establishing conversations would have less impact. Simulations with more conversations or shorter conversations were expected to favour ants-based proactive routing, since they would be able to use their existing routes rather than discover a new route whenever a conversation began.

Parameters	Values
Simulation Duration	900 sec
Number of Nodes	50
Node Maximum Movement Speed	5 m/s
Node Movement Pause Time	0, 60, 300 sec
Number of Movement Patterns	5
Test Area	1400m x 350m
Transmission Range	250m
Data Transmission Rate	2 Mbps
Number of Bi-Directional Conversations	5, 15, 25
Conversation Length	100, 200 sec
Conversation Interval	30, 60 sec
Packet Size	512 bytes
Packet Rate	4 packets/second

Table 4.1: A summary of the parameters used in my test simulations, and the values used for each parameter. Static parameters are included as well.

4.3 Recorded Results

Information recorded during a simulation run is used to evaluate the performance of the routing protocols. The central information for my evaluation is how successful a

routing protocol is at delivering packets to their destinations. This is found by tracking the number of packets transmitted and the number of packets actually delivered to their destinations.

The second important piece of information is how good a routing protocol is at getting packets to their destination in a timely manner. I monitored this by tracking the time each packet is transmitted and the time it is delivered, and taking the difference between the two. I also tracked the path length by monitoring the hop count field of packets when they are delivered to provide complementary information.

The final piece of data common to all routing protocols is how much of the network's transmission capacity it uses. This is measured by counting the number of routing packet transmissions each protocol performs. While the routing packets do vary in size, and the exact cost depends on how many other nodes are within radio range of a transmitting node, this gives a general indication of the overhead of a protocol.

For LANts and SLANts, I also recorded each node's estimate of ant lifetimes whenever these estimates change, and the actual ant lifetime whenever an ant was destroyed. No additional information was recorded for AODV.

Chapter 5

Results

5.1 Continuum Ants Algorithm Failure

My preliminary test simulations revealed a serious problem with CoAnts (the Continuum Ants Creation algorithm, described in Section 3.4). The ant population tended to spike abruptly or grow out of control, creating congestion. The causes of this behaviour are described in Section 3.4.3. Specifically, Equation 3.6 shows that the number of ants in the system must be increased when wait times increase. However, as more ants are added to the system, queues grow longer and, by queuing theory, the average wait time goes up asymptotically. This, in turn, causes an increase in the number of ants needed, continuing the cycle.

Figure 5.1 shows a comparison of the ant population using the Continuum Model-based population control algorithm and the population using the Lifetime-based control algorithm in a sample simulation. The graph clearly shows the difference in ant populations between the two algorithms. The dips in the continuum population occur when a large number of ants are destroyed at once. This happens when, for example, several ants are waiting in a queue for transmission to a single node and a transmission to that node fails. In this case, all of these ants are deleted at once, abruptly dropping the ant population.

Since the difference in routing protocol overhead is so massive and the continuum model did not demonstrate superior performance, no further tests involving it were run.

5.2 AODV, LAnts, and SLAnts Test Results

The main tests compared the AODV, Lifetime Ants (LAnts, see Section 3.5), and Snooping Lifetime Ants (SLAnts, see Section 3.6) routing protocols. The parameters

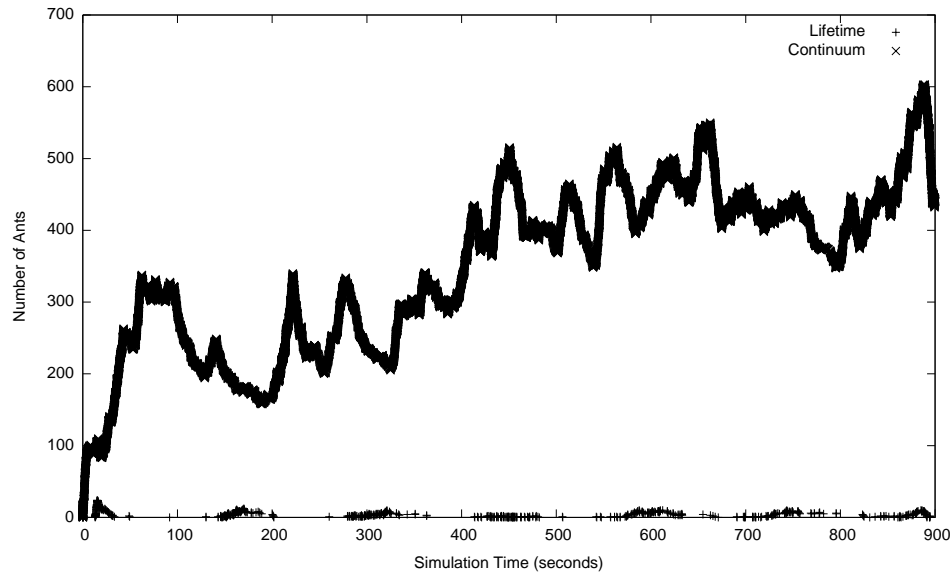


Figure 5.1: The number of ants present in a simulation run over 50 nodes using the Continuum Model-based population control algorithm, compared to the number of ants in a simulation run using the Lifetime-based algorithm.

for these tests and the reasoning behind the parameter selection are described in Chapter 4. I ran one test for each protocol for each combination of parameters, resulting in 180 tests in total. Since I used five different movement patterns for each pause time, this does not give enough results for any single combination of parameters to perform analysis of the statistical significance. All of the results in this section, and all of the conclusions presented later, are inferred from the tests that were performed, but more data is needed before conclusions can be drawn on the statistical significance of the results.

The primary metric used to compare the three algorithms is the number of packets successfully delivered by each routing algorithm. As shown by Figure 5.2, the percentage of packets delivered decreases as the number of two-way conversations increases for all routing protocols. This shows the performance of the SLAnts protocol to be superior to LAnts, which, in turn, is superior to AODV. AODV shows the most significant decrease as the number of two-way conversations increases, while lifetime and snooping ants show a more gradual decrease.

Figure 5.3, Figure 5.4, and Figure 5.5 show the percentage of packets delivered

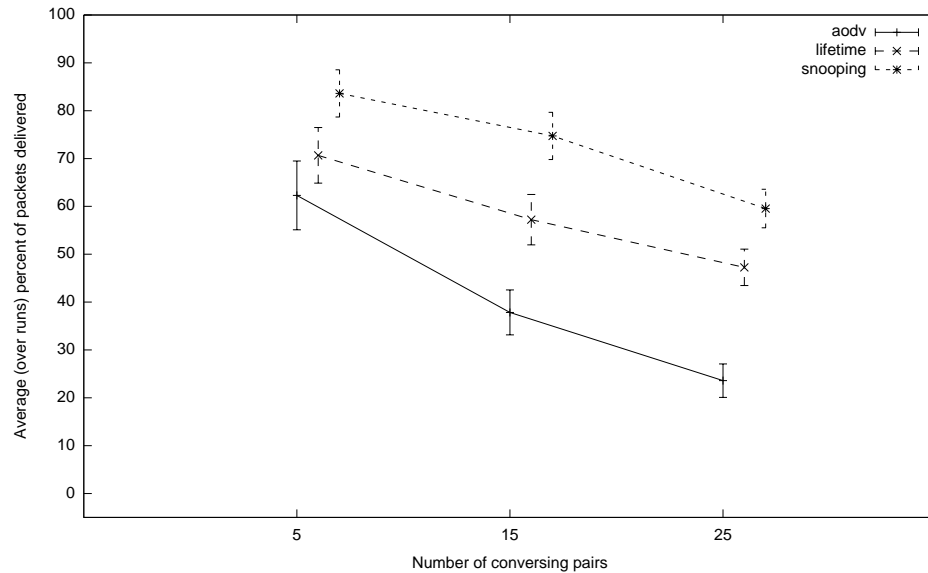


Figure 5.2: The percentage of packets delivered out of all transmitted data packets, averaged over time for each run then averaged across all runs for a given algorithm for each number of two-way conversations.

plotted against mobility for each number of conversations. The variation shown in Figure 5.2 means that the error bars are too large when runs with a different number of conversations are averaged together. The three protocols have the same relative ordering here, but for the most part, there is no variation with mobility. The exception is Figure 5.5, which shows the results for the case with 25 bi-directional conversations. Here, all protocols show a noticeable increase in successful deliveries as mobility decreases.

The second important metric is delivery delay, or how long it takes a packet to reach its destination once transmitted. Figure 5.6 shows that the LANts protocol has reduced delay compared to AODV, but the SLAnts protocol has much more delay than either LANts or AODV when there are 25 two-way conversations. The following graphs examine the 25 two-way conversation scenarios using other metrics and parameters, and a detailed examination and explanation is provided in Section 5.3.

Figure 5.7 shows that SLAnts' increased delay is drastically reduced in the highest-mobility scenario. In both of the scenarios with lower mobility, the delay is higher.

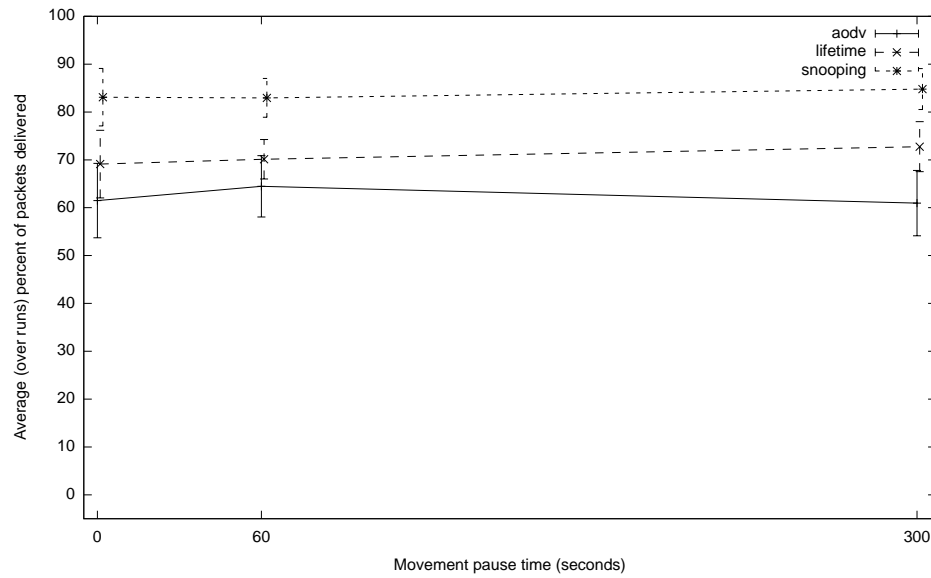


Figure 5.3: The percentage of packets delivered out of all transmitted data packets, averaged over time for each run then averaged across all runs with 5 conversations for a given algorithm for each movement pause time.

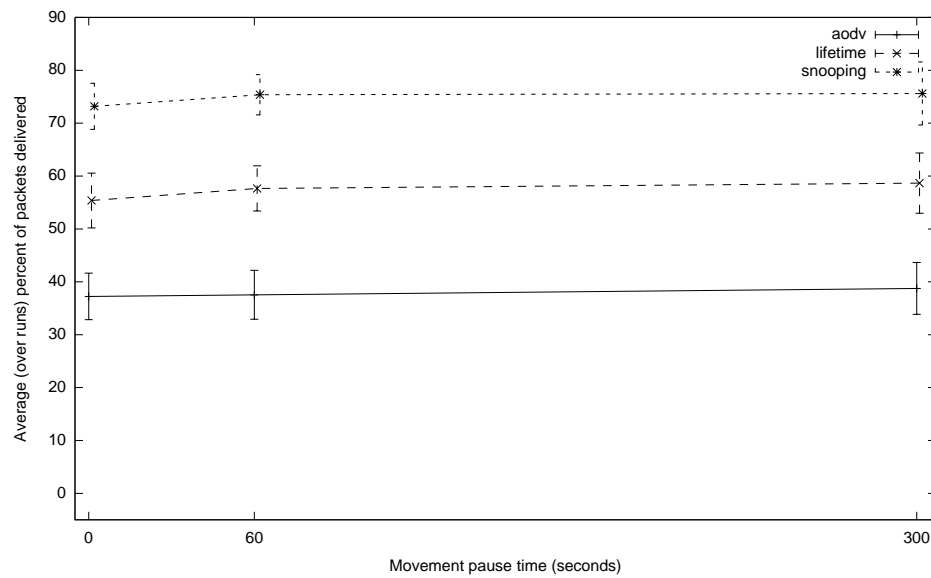


Figure 5.4: The percentage of packets delivered out of all transmitted data packets, averaged over time for each run then averaged across all runs with 15 conversations for a given algorithm for each movement pause time.

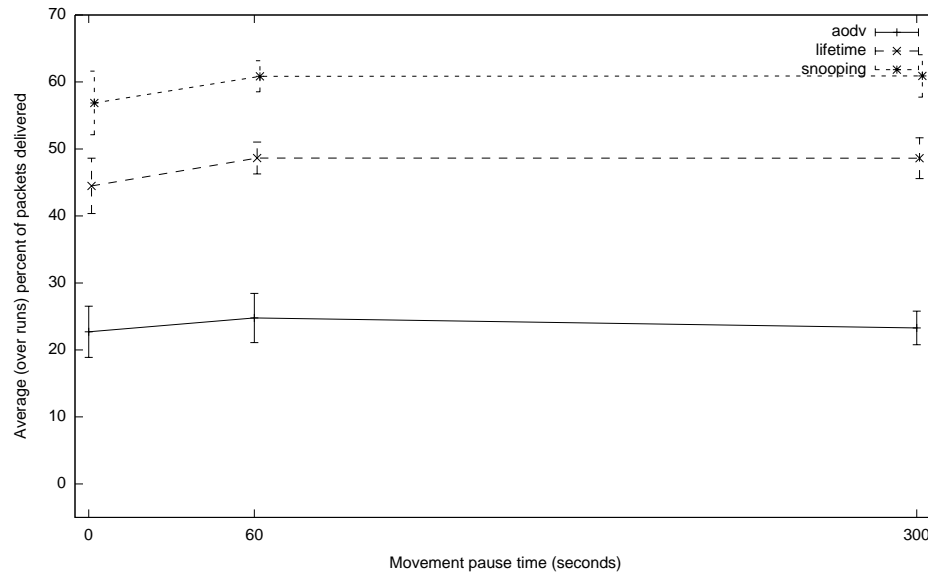


Figure 5.5: The percentage of packets delivered out of all transmitted data packets, averaged over time for each run then averaged across all runs with 25 conversations for a given algorithm for each movement pause time.

The delivery delay for SLAnts also has a much wider range, indicating that some scenarios are showing significantly increased delay times over the average. Both AODV and LAnts have smaller error bars and less variation in their delay times.

Figure 5.8 shows that longer conversations also result in increased delay for the SLAnts protocol. The result for 100 second conversations is much closer to the result for the other two protocols, and has a smaller error bar. The result for the 200 second conversations is not only much higher than that for the other two protocols, but again features much more pronounced error bars.

The last metric is the number of routing packets sent. For AODV, this is the number of times an RREQ packet is broadcast. For LAnts and SLAnts, this is the number of ant transmissions made. The two are directly comparable because of the similarity of broadcasts and unicasts in wireless networks, noted earlier in Chapter 1. Figure 5.9 shows that the number of routing packets transmitted by both LAnts and SLAnts remains mostly flat as the number of two-way conversations increases, while the number of Route Request broadcasts made by AODV increases. This indicates that the ants-based solution scales better with increased network activity.

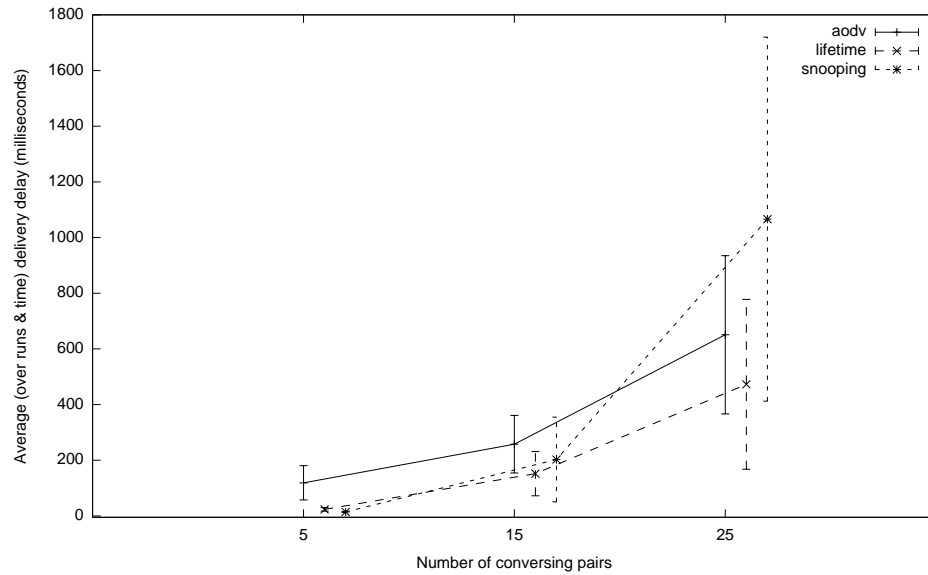


Figure 5.6: The delay time for data packets, averaged over time for each run then averaged across all runs for a given algorithm for each number of two-way conversations.

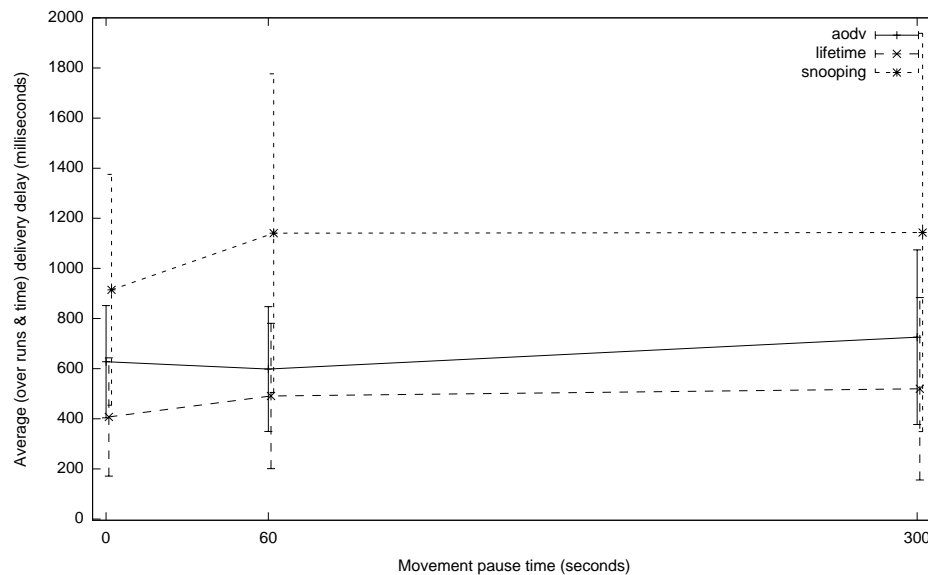


Figure 5.7: The delay time for data packets, averaged over time for each run with 25 conversation pairs then averaged across all runs for a given algorithm for each movement pause time.

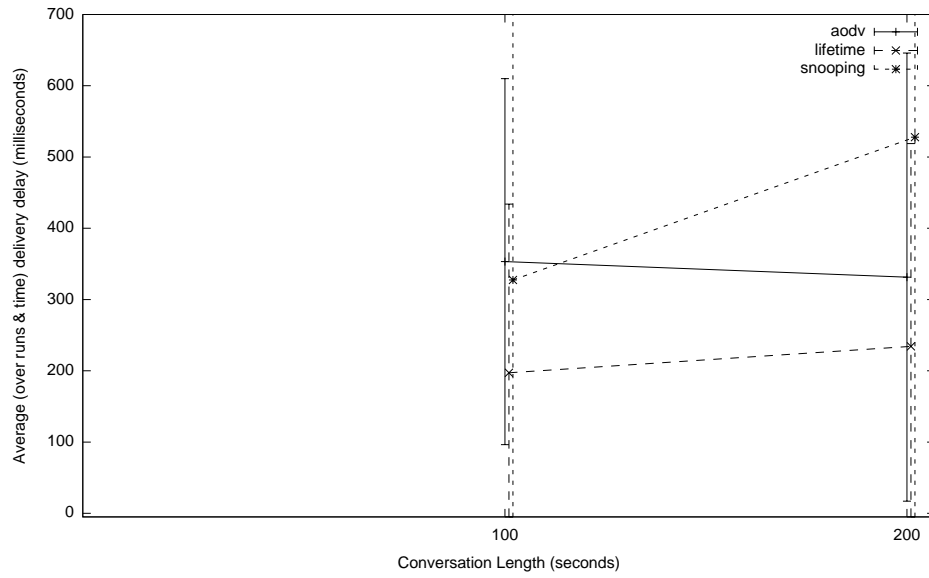


Figure 5.8: The delay time for data packets, averaged over time for each run with 25 conversation pairs then averaged across all runs for a given algorithm for each conversation length.

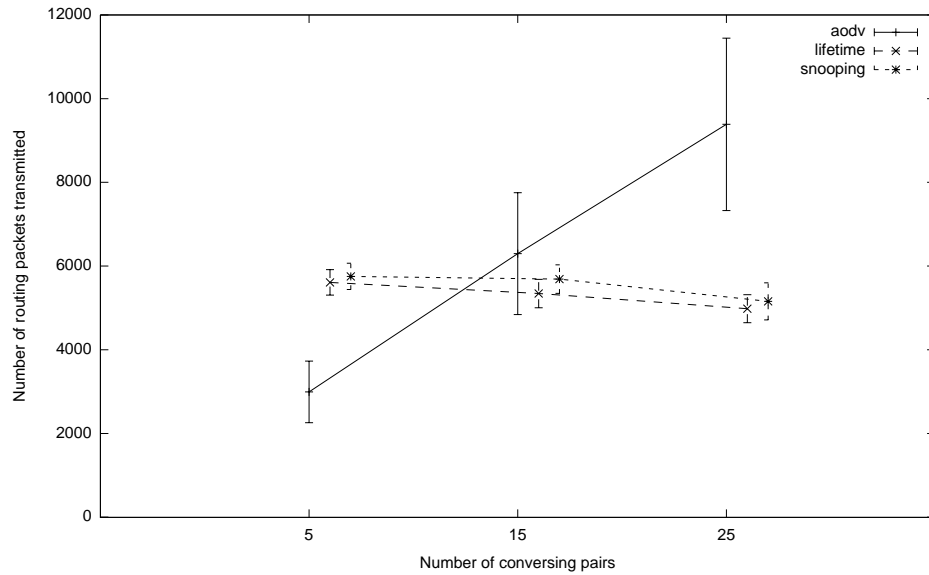


Figure 5.9: The number of routing packets transmitted (AODV Route Requests or LAnts/SLAnts ants) versus the number of two-way conversations.

Several parameter/metric comparison graphs are not shown here. The inter-conversation interval had little effect on the results. The graphs for all metrics for this parameter showed little or no variations, with the various protocols ordered the same as they were for the percentage of packets delivered. The conversation length parameter also had no influence on the percentage of packets delivered or the number of routing packet transmissions made. These graphs are included for reference in Appendix A.

5.3 Snooping Delay Analysis

The results described in Subsection 5.2 raise the question of why SLAnts suffers from such a large increase in delivery delay in high-traffic scenarios. To resolve this issue, I re-ran one of the tests that demonstrated a high delivery delay and examined the details of the simulation more closely. As Figure 5.10 shows, the snooping protocol seems to have drastically increased queue lengths over the other protocols. An examination of other log files revealed that this was not due to routing loops, longer routes, or the greater number of packets delivered.

Figure 5.11 shows the “weight” of the nodes in the network at a moment in time that was highly congested in the same test used to generate Figure 5.10. A node’s weight starts at 0 and is increased by one for every data stream that is routed through it or starts or ends at it. This graph was not conclusive, since an increase in weight of one node inevitably leads to the decrease in weight of another. However, the larger numbers of 1s, 5s, 6s, and 10s for SLAnts and the reduced number of 2s and 4s did suggest a direction for further investigation.

Examining the routing tables and statistics for this scenario closely yielded an explanation. Sequence numbers prioritize new information very strongly. No matter how good a node’s existing route to a destination is, if routing information with a higher sequence number is received, it is used in place of the old information. When an ant passes through a node in ordinary LAnts, all nodes it passes through use its route to that node. SLAnts makes this even worse, because all nodes within radio range of the path the ant takes use this new information. New routes quickly dominate the network, meaning that the entire network is using the same path along an ant’s route

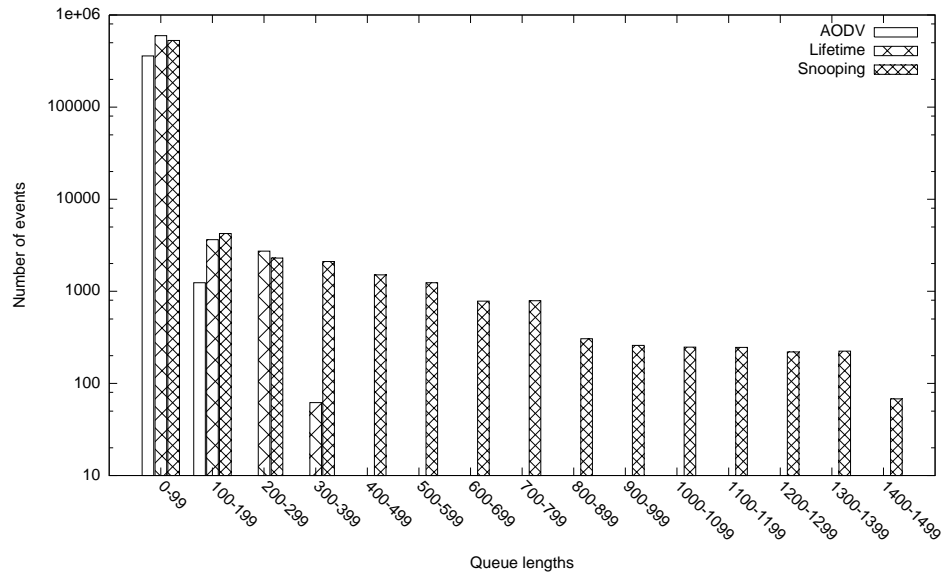


Figure 5.10: A histogram of queue lengths for different protocols. Queue lengths were sampled at random intervals over the course of the simulation.

to communicate with a node.

In a high-traffic scenario, this can result in a small handful of nodes routing a large amount of traffic. Without snooping, this traffic could have been distributed over several routes of different ages.

5.4 Lifetime Estimation Accuracy

While Section 5.2 has established the overall effectiveness of the LAnts protocol, it does not address the question of the accuracy of the lifetime estimates used to choose ant creation intervals. Examining the log files reveals that the ant lifetime estimates are very accurate, and quickly adjust to reflect changes in ant lifetimes.

The networks used in these scenarios are small enough that lifetime estimates do not vary widely between nodes. This can be seen in the lifetime ant populations in Figure 5.1. The clustered periods of heightened ant population are caused by all of the nodes starting at the same time, and all having similar lifetime estimates. The length and variation are caused by both delay and overall randomness. In the “empty” periods between clusters of ants, the overall ant population is zero.

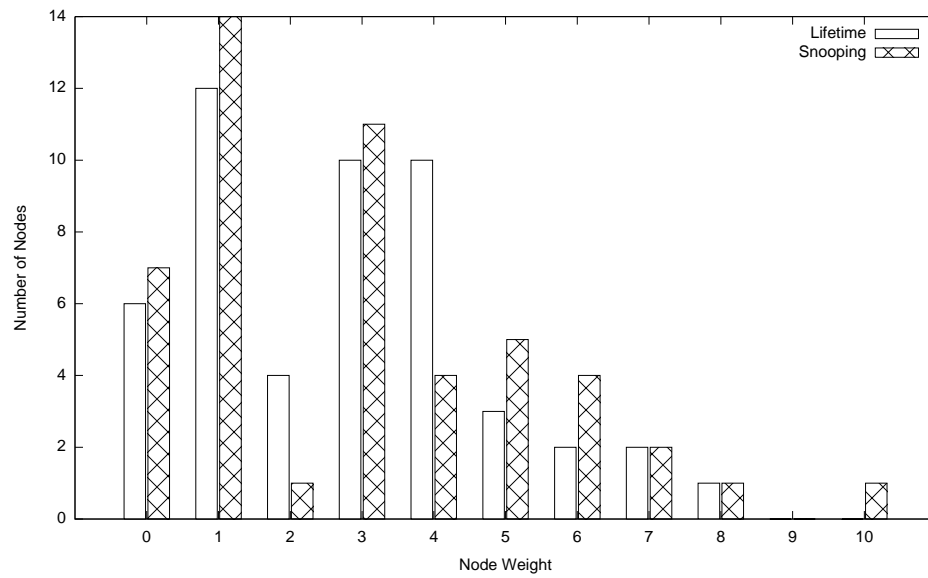


Figure 5.11: A histogram of node weights for lifetime and snooping protocols from a snapshot of routing tables taken at a point of high congestion during a test.

Chapter 6

Conclusions

In this thesis, I have developed and presented two new MANET routing protocols: LAnts and SLAnts. LAnts builds on previous ant-based routing protocols from Zhou [31], Yue [30], and Thota [27] by adding a distributed ant population control algorithm that operates based only on local information. SLAnts extends LAnts by using snooping on both data packets and routing packets to improve routing data collection. Both protocols were tested using a discrete event simulator, and the results are compared against each other and against AODV, a standard MANET routing protocol.

My simulations exercised all three routing protocols in a variety of circumstances. Both mobility and data traffic were varied in several different ways. This allowed the protocols to be tested for both their response to changes in network structure and their behaviour under heavy traffic loads. While not enough tests were run for any single combination of protocols to allow for analysis of statistical significance, the results still provide an indication of the overall performance of the protocols.

LAnts and SLAnts both show improvement over the standard AODV MANET routing protocol in terms of packets delivered. They maintain this advantage under both heavy traffic load and high mobility. The improved packet delivery demonstrated by LAnts and SLAnts does have an attached cost. When overall network traffic is low, both LAnts and SLAnts have a higher overhead than AODV because they still attempt to maintain routes between all nodes, instead of focusing on nodes actively engaged in communication. When network traffic becomes heavier, AODV's bandwidth consumption increases drastically, while LAnts and SLAnts remain at the same level.

Similarly, LAnts has an improved delivery delay time over AODV. SLAnts does not share this trait, and has a worse delivery delay time than either LAnts or AODV in

the high-traffic scenarios with 25 bi-directional conversations. The sequence number mechanism used to prevent routing loop formation is responsible for this increased delay, since it heavily favours newer routes. Because SLAnts' snooping allows it to disseminate routing information more rapidly than LAnts, these new routes are spread much more rapidly. LAnts' slower information dissemination works to its advantage here, because many nodes are still using old routes, spreading out data traffic between several nodes instead of concentrating it through one.

6.1 Future Work

One thing that can be done to build on the work presented in this thesis would be to run more tests on the existing algorithms. Running enough tests with each set of parameters to analyse the statistical significance of the results is important. Tests should also be run with different parameters and scenarios, to test the protocols in a wider variety of situations. A different method of controlling network mobility would be particularly useful, as movement pause time seems to have had little effect. LAnts and SLAnts should also be compared against other MANET routing protocols, including DSDV, DSR, ZRP, OLSR, and other insect-metaphor routing systems, including AntHocNet and bee-based MANET routing.

The foremost problem revealed in the tests I ran is the jump in the delay time under SLAnts. There are several possible solutions that may be worth investigating, including replacing the sequence numbers with another method for eliminating routing loops and prioritizing shorter paths over newer paths as long as the route has not been marked invalid. All of the standard mechanisms for avoiding or correcting routing loops should be considered, including split horizon, poison reverse, and count to infinity. Multi-path routing could also improve the performance of this algorithm, avoiding the congestion at one node by keeping several valid paths to a destination and choosing randomly between them.

Another area that may have potential for further work is protocol start-up. LAnts currently initializes the average ant lifetime to zero, and relies on the lower bound from the link layer timeout (described in Section 3.5.1) to prevent a flood of ants. The protocol still produces a large but manageable number of ants here, and quickly

develops its normal operation patterns, but this could be an issue with a node joining an already-existing network.

Further refinement to the use of snooping in SLAnts is also possible. As was mentioned in Section 3.6, snooping on data packets could be used as a replacement for hello messages. Using this, a node could hold off on sending a hello message as long as it has transmitted a data packet recently. Another potential development is further breaking down the barriers between routing packets and data packets. The IPv6 extension header mechanism could be used to attach a small ant to every data packet, allowing it to carry updated routing information along its path to its destination and providing even more data for snooping nodes. The IPv4 record route option could also be used to record this information.

X , the ant population scaling constant used in both LAnts and SLAnts, is another potential subject for future work. The tests use a constant X , but varying X in response to changes in network connectivity, mobility, traffic level, or congestion might improve protocol performance, either by increasing the number of packets delivered or decreasing the number of ants sent in low-traffic scenarios.

Examining how the ant lifetime estimates propagate in larger ad-hoc networks and how this relates to the ant random walk mechanics developed by Thota [27] would provide useful insight into the operation of the protocols. Larger networks should demonstrate some variation in estimated lifetimes due to differing local conditions, but how these differences affect routing and population control behaviour is unclear.

The ant population control mechanics used by LAnts and SLAnts are primarily concerned with regulating the ant population in response to network congestion and activity. They do not make any attempt to adjust the ant population in response to mobility, and instead allow routing performance to degrade as mobility increases. Adding mobility as a factor in determining the ant creation interval may improve the protocol's performance. Along the same lines, it may be possible for SLAnts to increase the ant creation interval to account for the extra information provided by snooping.

Bibliography

- [1] The network simulator, NS2. <http://www.isi.edu/nsnam/ns/>.
- [2] Ozalp Babaoglu, Geoffrey Canright, Andreas Deutsch, Gianni A. Di Caro, Frederick Ducatelle, Luca M. Gambardella, Niloy Ganguly, Márk Jelasity, Roberto Montemanni, Alberto Montresor, and Tore Urnes. Design patterns from biology for distributed computing. *ACM Trans. Auton. Adapt. Syst.*, 1(1):26–66, 2006.
- [3] M. Benzaid, P. Minet, and K. Agha. Integrating fast mobility in the olsr routing protocol, 2002.
- [4] IEEE-SA Standards Board. IEEE standard 802.11, 1999 edition. IEEE Standards for Local and Metropolitan Area Networks: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, June 2003.
- [5] Josh Broch, David A. Maltz, David B. Johnson, Yih-Chun Hu, and Jorjeta Jetcheva. A performance comparison of multi-hop wireless ad hoc network routing protocols. In *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*, pages 85–97, New York, NY, USA, 1998. ACM Press.
- [6] T. Clausen and P. Jacquet. Request for comments: 3626, optimized link state routing protocol (OLSR). <http://www.ietf.org/rfc/rfc3626.txt>, October 2003.
- [7] B. Ducourthial, Y. Khaled, and M. Shawky. Conditional transmissions, a strategy for highly dynamic vehicular ad hoc networks. In *IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks*, pages 1–8. IEEE, 2007.
- [8] Michael Feely, Norman Hutchinson, and Suprio Ray. Realistic mobility for mobile ad hoc network simulation. In *Ad-Hoc Mobile and Wireless networks*, volume 3158 of *Lecture Notes in Computer Science*, pages 324–329. Springer Berlin / Heidelberg, 2004.
- [9] Ying Ge, T. Kunz, and L. Lamont. Quality of service routing in ad-hoc networks using olsr. In *Proceedings of the 36th Annual Hawaii International Conference on System Sciences*, 2003.
- [10] Tom Goff, Nael B. Abu-Ghazaleh, Dhananjay S. Phatak, and Ridvan Kahvecioglu. Preemptive routing in ad hoc networks. In *MobiCom '01: Proceedings of the 7th annual international conference on Mobile computing and networking*, pages 43–52, New York, NY, USA, 2001. ACM Press.

- [11] Ernst W. Grundke. Routing table dynamics in mobile ad-hoc networks: A continuum model. In *Proceedings of the 3rd International Conference on Ad-Hoc Networks and Wireless*, volume 3158 of *Lecture Notes In Computer Science*, pages 318–323. Springer-Verlag / Heidelberg, 2004.
- [12] Ernst W. Grundke and A. Nur Zincir-Heywood. A uniform continuum model for scaling of ad-hoc networks. In *Proceedings of the 2nd International Conference on Ad-Hoc Networks and Wireless*, volume 2865 of *Lecture Notes in Computer Science*, pages 96–103. Springer-Verlag / Heidelberg, 2003.
- [13] Zygmunt J. Haas, Marc R. Pearlman, and Prince Samar. The zone routing protocol (ZRP) for ad hoc networks. <http://people.ece.cornell.edu/%7Ehaas/wnl/Publications/draft-ietf-manet-zone-zrp-04.txt>, July 2002.
- [14] Per Johansson, Tony Larsson, Nicklas Hedman, Bartosz Mielczarek, and Mikael Degermark. Scenario-based performance analysis of routing protocols for mobile ad-hoc networks. In *MobiCom '99: Proceedings of the 5th annual ACM/IEEE international conference on Mobile computing and networking*, pages 195–206, New York, NY, USA, 1999. ACM Press.
- [15] D. Johnson, Y. Hu, and D. Maltz. Request for comments: 4728, the dynamic source routing protocol (DSR) for mobile ad hoc networks for IPv4. <http://tools.ietf.org/html/rfc4728>, February 2007.
- [16] Christine E. Jones, Krishna M. Sivalingam, Prathima Agrawal, and Jyn Cheng Chen. A survey of energy efficient network protocols for wireless networks. *Wireless Networks*, 7:343–358, 2001.
- [17] Michael Langhans, Constantin Timm, and Sebastian Werner. Beead hoc linux implementation of bee inspired routing algorithm. <http://mila.landlos.de/beeadhoc.pdf>, September 2005.
- [18] Tony Larsson and Nicklas Hedman. Routing protocols in wireless ad-hoc networks - a simulation study. Master's thesis, Luleå University of Technology, 1998.
- [19] Sung-Ju Lee, Mario Gerla, and Chai-Keong Toh. A simulation study of table-driven and on-demand routing protocols for mobile ad hoc networks. *IEEE Network*, 13(4):48–54, Jul/Aug 1999.
- [20] 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 *Proceedings of the 6th Annual International Conference on Mobile Computing and Networking*, pages 120–130, Boston MA, August 2000.

- [21] Nelson Minar, Kwindla Hultman Kramer, and Pattie Maes. *Cooperating Mobile Agents for Dynamic Network Routing*, chapter 12. Springer-Verlag, 1999.
- [22] V. Park and S. Corson. IETF MANET working group draft: Temporally-ordered routing algorithm (TORA) version 1 functional specification. <http://www3.ietf.org/proceedings/02mar/I-D/draft-ietf-manet-tora-spec-04.txt>, July 2001.
- [23] C. Perkins, E. Belding-Royer, and S. Das. Request for comments: 3561, ad hoc on-demand distance vector (AODV) routing. <http://tools.ietf.org/html/rfc3561>, July 2003.
- [24] C. E. Perkins, E. M. Royer, S. R. Das, and M. K. Marina. Performance comparison of two on-demand routing protocols for adhoc networks. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, volume 1, pages 3–12. IEEE, 2000.
- [25] 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, 1994.
- [26] Prasun Sinha, Raghupathy Sivakumar, and Vaduvur Bharghavan. Enhancing ad hoc routing with dynamic virtual infrastructures. In *Proceedings Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1763–1772, 2001.
- [27] Sessa Rao Thota. A performance model for an agent-based routing protocol for manets. Master’s thesis, Dalhousie University, December 2005.
- [28] Horst F. Wedde, Muddassar Farooq, Thorsten Pannenbaecker, Bjoern Vogel, Christian Mueller, Johannes Meth, and Rene Jeruschkat. Beeadhoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior. In *GECCO ’05: Proceedings of the 2005 conference on Genetic and evolutionary computation*, pages 153–160, New York, NY, USA, 2005. ACM Press.
- [29] Shih-Lin Wu, Sze-Yao Ni, Yu-Chee Tseng, and Jang-Ping Sheu. Route maintenance in a wireless mobile ad hoc network. *Telecommunications Systems*, 18(1-3):61–84, 2001.
- [30] Wenwei Yue. An improved agent-based routing protocol for mobile ad-hoc networks. Master’s thesis, Dalhousie University, July 2004.
- [31] Yan Zhou. Intelligent agent routing for mobile ad-hoc networks. Master’s thesis, Dalhousie University, April 2003.

Appendix A

Other Figures

These figures were not included in the main body of the thesis because they were uninteresting. They are provided here for completeness. Some figures in this section show some variation, but the scope of the variation is much smaller than the error bars.

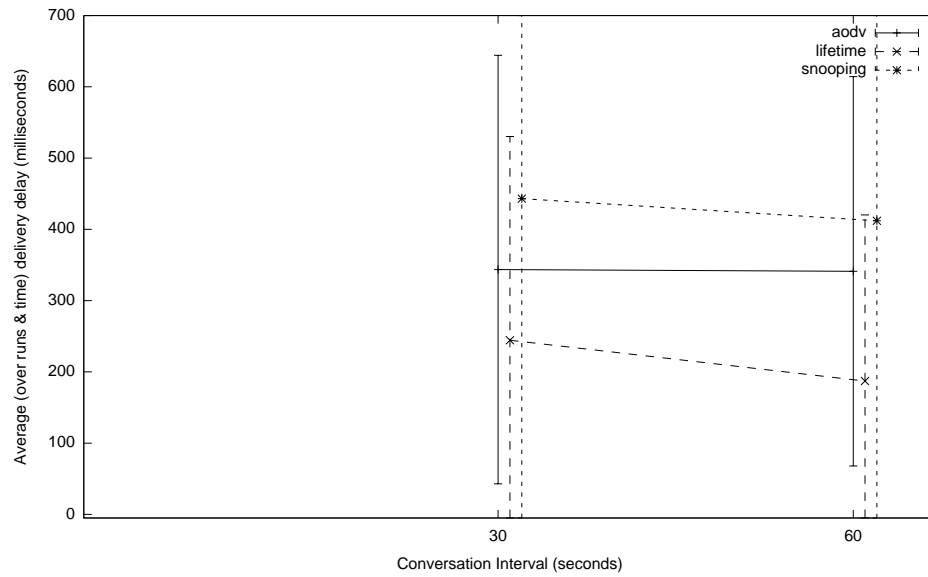


Figure A.1: The delay time for data packets, averaged over time for each run then averaged across all runs for a given algorithm for each conversation interval.

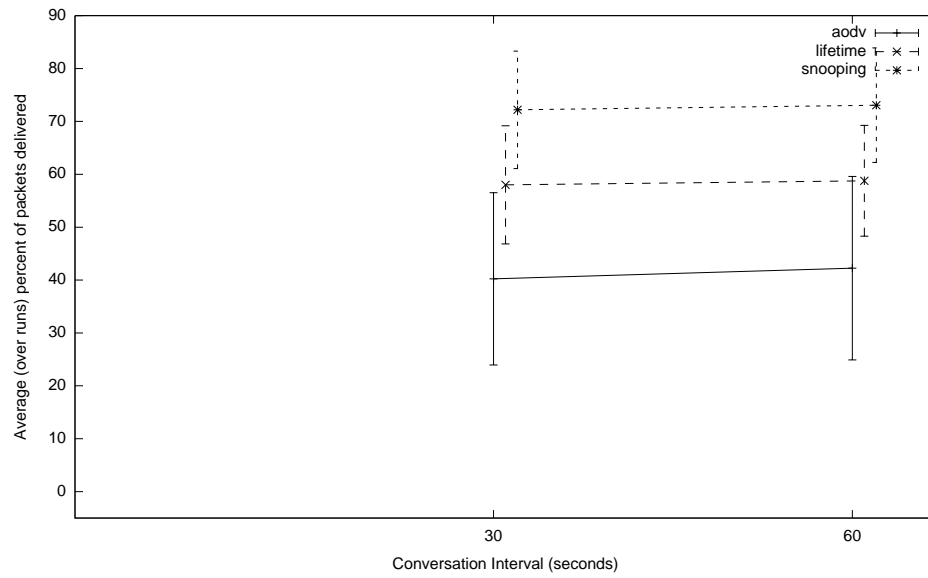


Figure A.2: The percentage of packets delivered out of all transmitted data packets, averaged over time for each run then averaged across all runs for a given algorithm for each conversation interval.

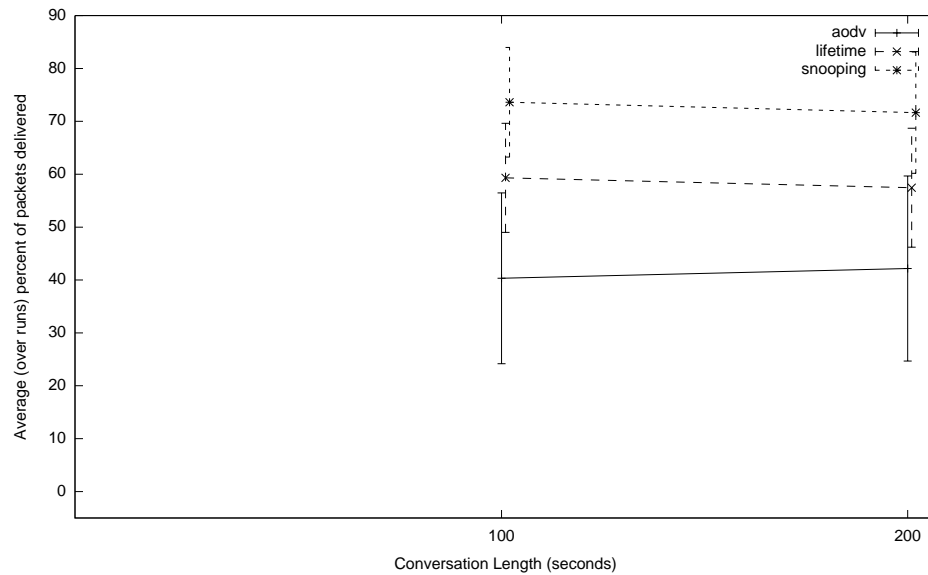


Figure A.3: The percentage of packets delivered out of all transmitted data packets, averaged over time for each run then averaged across all runs for a given algorithm for each conversation length.

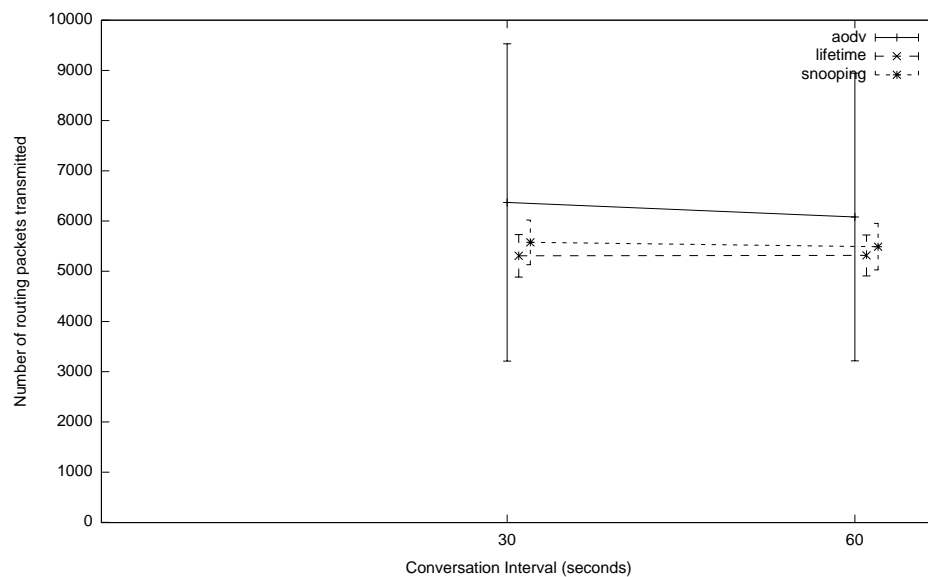


Figure A.4: The number of routing packets transmitted (AODV Route Requests or LAnts/SLAnts ants) for each conversation interval.

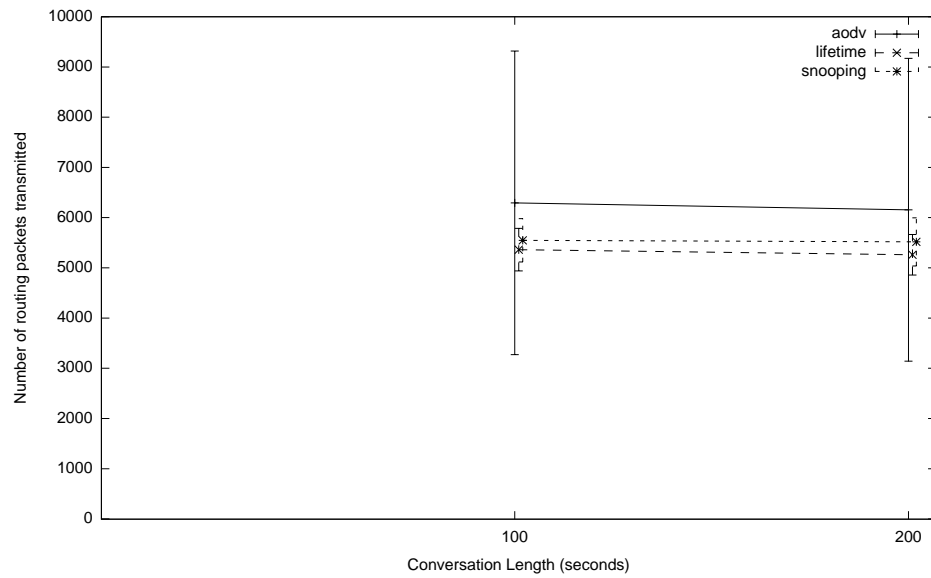


Figure A.5: The number of routing packets transmitted (AODV Route Requests or LAnts/SLAnts ants) for each conversation length.

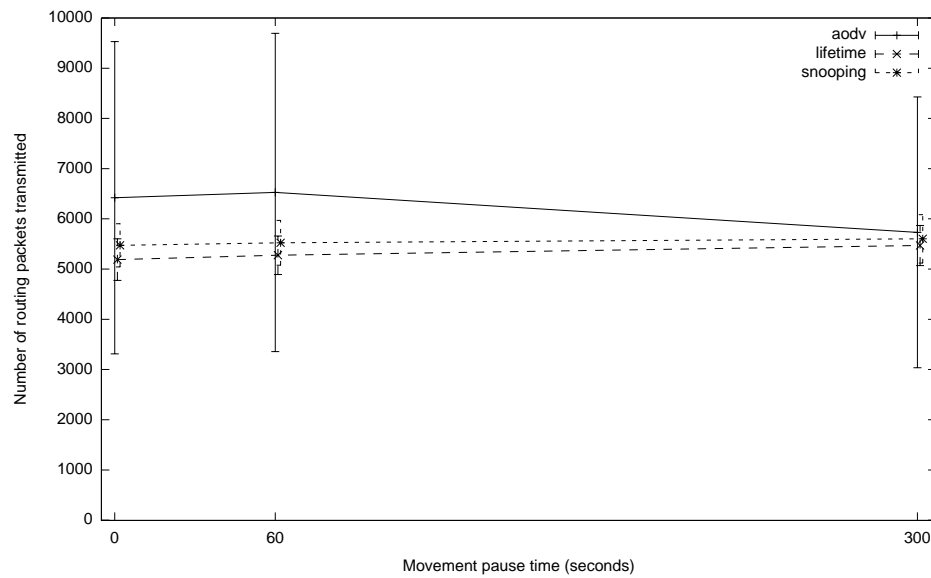


Figure A.6: The number of routing packets transmitted (AODV Route Requests or LAnts/SLAnts ants) for each movement pause time.