HISTORY-BASED ROUTE SELECTION FOR REACTIVE AD HOC ROUTING PROTOCOLS

By

PETER MICHAEL CAPPETTO

A thesis submitted in partial fulfillment of
the requirements for the degree of

MASTER OF SCIENCE

WASHINGTON STATE UNIVERSITY
School of Electrical Engineering and Computer Science

MAY 2007

To the Faculty of Washington State University:

The members of the Committee appointed to examine the thesis of PETER MICHAEL CAPPETTO find it satisfactory and recommend that it be accepted.

_____
Chair

_____

_____

ACKNOWLEDGEMENT

I would like to thank all the people who have helped me along my way towards completing my master's degree. First and foremost, I would like to thank Dr. Sirisha Medidi for her support and guidance. I am greatful towards Dr. Murali Medidi and Dr. Carl Hauser for their participation on my committee. Most of all, I would like to thank my family for all their support.

HISTORY-BASED ROUTE SELECTION FOR REACTIVE AD HOC ROUTING PROTOCOLS

Abstract

by Peter Michael Cappetto, M.S.
Washington State University
May 2007

Chair: Sirisha Medidi

Mobile ad-hoc networks support a wide variety of applications such as battlefield surveillance, environmental monitoring, emergency response, to name a few. These networks are fundamentally characterized by their wireless communication medium and are resource-constrained in terms of size, bandwidth and infrastructure. The network topology may change rapidly and unpredictably over time. The network is maximally decentralized, where all network activity including route discovery and message transmission must be executed and coordinated by the nodes themselves. Challenges in deploying these networks include reliable time-critical information for decision-making and secure communication. Developments in ad hoc networking in these areas would greatly enhance our capability to effectively deploy them in emergency and critical conditions.

Ad hoc networks rely on cooperation, but selfish nodes drop packets to preserve their battery life. This behavior degrades the network performance; judicious route selection will maintain and improve the network performance in the presence of selfish or misbehaving nodes. Current routing algorithms choose routes based on shortest paths rather than the reliability of the path. This thesis proposes a light-weight route selection algorithm, history based route selection that uses past behavior to judge the quality of a route rather than solely on the length of the route.

History-based route selection draws information from the underlying routing layer at no extra cost and selects routes with a simple algorithm. This technique maintains the node's history in a

small table, which does not place a high cost on memory. History-based route selection's minimalism suits the needs the portable wireless devices and is easy to implement. Since the routes are chosen based on past behavior, this technique provides stable routes and improved packet reception. The algorithm was implemented and tested using the ns-2 simulator. Simulation results show that history-based route selection achieves higher packet delivery and improved stability than its length-based counterpart.

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

Dedication

I would like to dedicate this to my parents. Their support and encouragement has been invaluable.

CHAPTER ONE

INTRODUCTION

Ad hoc networks function without pre-existing infrastructure and do not have centralized administration. This freedom permits a wide range of novel uses from battlefield coordination to disaster response. Ad hoc networks also encompass exciting areas like sensor networks [10, 2] and opportunistic networks [24, 23, 1]. Ad hoc networks rely on cooperation to operate. In many ad hoc networks, the participating devices face resource constraints such as limited battery life and processing power. This altruistic behavior does not directly benefit intermediate nodes acting as forwarders; on the contrary, they expend their own resources on behalf of other nodes. Intermediate nodes can save their own resources by acting selfishly and dropping packets. Selfishness and other forms of packet loss hurts the network performance.

This problem is important because selfish behavior degrades the network performance in a variety of forms [33]. In reliable data transmission protocols, packet drops lead to retransmissions of the dropped packet. Packet dropping also leads to decreased performance at the routing layer. When DSR, a reactive ad hoc routing protocol, experiences a packet loss, it removes all the routes from its cache and must launch a new route discovery. The route discovery process sends request packets by flooding the network, which naturally contributes network overhead. The wireless environment contributes natural sources of packet loss, so misbehavior related drops only complicates an already difficult environment.

Nodes drop packets for many reasons. Congested nodes run out of buffer space and must drop packets when their queues becomes full. Malfunctioning nodes drop packets because of software or hardware problems. In hostile environments enemy nodes actively drop packets through attacks such as the blackhole or grayhole attacks [15, 21]. Spurious packet drops leads to decreased network performance. Judicious route selection accounting for misbehavior helps to mitigate the effects of misbehaving nodes as well as preserve resources.

Popular routing algorithms like Dynamic Source Routing(DSR) [19, 20] and Ad hoc On Demand Distance Vector(AODV) [28, 30] do not consider the reliability of the route when choosing a path for a packet. Route selection metrics like the shortest path do not consider the reliability of the route. This cannot be said of nodes in a network with malicious, malfunctioning, or selfish nodes. The route selection algorithm metric influences the performance of the nework.

Past performance of a node provides a history-based metric. A node that consistently forwarded packets correctly has more utility than a node that has dropped many packets. Given a short, error-prone route and a longer, more reliable route, a packet sent along the longer route has a greater delay, but the packet may have a higher chance of reaching the destination. Source nodes using history gather, store, and evaluate the quality of their potential forwarders.

Direct observation of the misbehaving node's behavior helps source nodes identify offending nodes. Because wireless is an open medium, nodes are able to promiscuously listen to their neighbors' transmissions and detect misbehavior in their one-hop neighborhood. Communication protocols also provide insight into node behavior. Using a protocol with acknowledgments lets the sender know that their packet arrived safely at the destination given; unacknowledged traffic does not have this luxury. Acknowledgments also show the nodes on that path have cooperated and attest their reliability. Other protocol information can also reflect the quality of nodes, for example, ICMP Destination Unreachable messages. Source nodes need not rely purely on their own ratings; they can incorporate outside knowledge.

Second-hand information distribution, termed gossiping, allows a node to use other node's ratings. A node will periodically distribute its assessment of others. The receiving nodes can incorporate these values with their own to form a broader picture of network behavior. Using second-hand information introduces some problems. Second-hand information increases the overhead due to the periodic rating distribution. Unscrupulous nodes can publish false values and spread bogus values to other nodes. Adversarial nodes can collude and boost each other's ratings when in fact they are misbehaving.

Source node can take action to deal with misbehaving nodes. After observing the node and evaluating it based on the misbehavior criteria, the system flags the node as misbehaving. Many systems set a threshold value on their evaluation metric and flag the node when it exceeds that value [25, 7]. Routes with the node can be removed entirely, or receive a low rating. Not all misbehaving nodes are malicious and may start to function correctly later. At this point, the rating system can begin a forgiveness process and allow the node to redeem itself from its past behavior.

We propose a light-weight route selection method that only relies on information received from the routing layer. This approach only uses first-hand information and and does not require the participation of other nodes in misbehavior management. This follows the reasoning that not all nodes in the network cannot be trusted to run non-modified copies of the routing algorithm. History-based routing takes a light-weight approach and does not require intense computation, which suites the needs of energy constrained devices. It uses freely available routing information to base decisions and does not require significant modification to the underlying routing algorithm. Our algorithm uses a small amount of memory per node to maintain a record of past behavior. We show that this method performs well under high mobility conditions.

The thesis has the following structure. Chapter 2 provides a brief background useful understanding the problem domain. Chapter 3 introduces related work coming from differing perspectives. We describe our approach, history-based route selection, in chapter 4 and illustrate the details behind the algorithm. Chapter 5 explains the goals behind experiments used to test the solution. It details and justifies the metrics used and explains the results. Chapter 6 presents our conclusions and future work.

CHAPTER TWO

BACKGROUND

This chapter covers background material relevant to ad hoc routing. It gives an overview of the characteristics of ad hoc networks, factors influencing routing protocol design, and a discussion of the Dynamic Source Routing protocol. This helps lay the groundwork for subsequent chapters.

2.0.1 Ad Hoc Networks

Ad hoc networks often consist of wireless devices, many of which are portable. Unlike devices like desktop computers, mobile devices rely on battery technology. This limits the lifetime of the device, but software running on these devices mindful of this can help to extend the life. Portable devices typically have less powerful hardware than their fixed counterparts. This is especially true of nodes in sensor networks, which are a specialized ad hoc network even more constrained by limited device resources.

The network structure changes with time as new nodes enter, old nodes leave, and current nodes move. Wired routing protocols do not translate well to this environment, so researchers have proposed a variety of new techniques mindful of the ad hoc network environment. Mobility gives users freedom to move about, but it also creates additional challenges for the routing protocol. Route discovery and route maintenance are two main issues addressed by ad hoc routing protocols. Route discovery entails obtaining routes to a given destination. The route maintenance process helps remove routes that no longer exist.

2.0.2 Ad Hoc Routing Protocols

Ad hoc routing protocols follow two major designs: reactive and proactive. Proactive routing protocols issue periodic messages that contain relevant network information. These protocols often store the data in a route table. When a sender wants to obtain a route to a particular destination, it consults the table. This has the advantage of quick route lookups, but it introduces the problem

of periodically broadcasting network information. These broadcasts increase the overhead and generate traffic even without active connections. Proactive routing protocols do not scale very well, which limits their use. Destination-Sequenced Distance-Vector Routing [29] is one example of a proactive routing protocol.

Reactive routing protocols only obtain routes to destinations on demand. They do not create traffic unecessarily unlike proactive protocols. These protocols discover routes through the route query process. This often results in a node broadcasting a request for a route to the destination. Broadcasted requests propagate throughout the network until the destination receives the request or the packets are dropped for exceeding some number of hops.

2.0.3 Dynamic Source Routing

The DSR protocol is a popular source routing protocol. DSR is a reactive protocol that has small overhead and scales well. It has three main control packets:

- Route Request (RREQ)
- Route Reply (RREP)
- Route Error (RERR)

DSR uses route request packets to build a source route. If a node does not already have a route to a destination in its cache, then it broadcasts a route request packet. Intermediate nodes receiving the packet first look at the request id enclosed in the packet to determine if they have already received the request. If they have already processed the request, then they silently discard the packet. If the request is new, then the intermediate node places its address within the packet and broadcasts the packet to its neighbors. The packet continues to be broadcasted throughout the network collecting addresses. The destination processes the first such request to arrive. This is based on the assumption that the first packet to arrive has come from the shortest path.

The destination creates a route reply packet and attempts to send it back to the requesting node. If the destination already has a route to the sender, then it sends the packet back along that path. DSR does not presume symmetric links, so the receiver must initiate its own route request if it does not have a path back ot the source. The packet adds the route to its route cache and sends any buffered packets that were waiting for a route.

The route maintenance process helps nodes to remove broken routes from their caches. Intermediate nodes create route error packets when they cannot forward the packet to the next hop along the route. They send this packet back to the sender alerting them of the broken link. The receiver of such an error packet removes all the cached routes containing the node where the error occured. Intermediate nodes forwarding or overhearing the packet may also process the packet and remove the faulty routes. Routes also become stale if they have not been used in a certain period of time. Nodes purge expired routes from their caches in favor of obtaining a more up to date route.

CHAPTER THREE

RELATED WORK

In this chapter we examine related work. Since many solutions to network problems for wired networks do not often translate well to a wireless ad hoc environment, a new set of solutions is required. Related misbehavior mitigation methods tackle the problem from different perspectives, but they share the common goal of maintaining good network performance in the presence of misbehaving nodes.

3.1 Reputation Systems

Reputation systems assign nodes reputations based on their behavior. Designers must decide what set of behaviors they are interested in observing and how to observe it. In many cases dropping a packet constitutes misbehavior, but other actions indicate misbehavior as well. For example, altering the packet contents is also considered a malicious act. Nodes collect this data based on their observations and create an assessment of the other nodes. They may also incorporate the reputation values that other nodes publish. The route selection algorithm then chooses routes based on the stored reputation values.

3.1.1 Watchdog and Pathrater

Marti et al. [25] describe a misbehavior detection system that uses promiscuous listening to determine whether or not the next hop along a route has forwarded a packet correctly. Their system alerts source nodes when an intermediate node misbehaves, which allows the source node to avoid the offender in subsequent route selections. Promiscuous listening ensures that the next hop neighbor does not tamper with the contents of a packet.

Their system consists of two components: the watchdog and the pathrater. The watchdog stores packets and listens promiscuously for the next hop neighbor to forward the packet. It compares the transmission to the buffered copy of the packet and detects alteration of the packet. If a neighboring

node exceeds a threshold value for dropped packets, then it becomes flagged as misbehaving. Intermediate nodes that have flagged another node as misbehaving send a warning message back to the source to alert them. The pathrater component maintains ratings of nodes and chooses routes based on these values. Paths containing misbehaving nodes have a large negative weight, which greatly reduces the attractiveness of the route to the selection algorithm.

Their simulation efforts involved running experiments varying the active components, the watchdog and the pathrater, to establish how much each of the components affected the performance. The authors' results showed an improvement in the over all network throughput and a similar level of overhead with all components active. The case with all components on also had the highest packet delivery with at all levels of misbehavior.

This approach performs well for its simplicity, but it also has limitations. The authors discuss the shortcomings of their system such as when a node listening for the next hop transmission experiences collision due to the transmission from another neighbor. Even though the next hop forwarded the packet, the watchdog component falsely attributes poor behavior to that node. Others have criticized it for not punishing the offenders [7, 11]. Properly behaving nodes tasked with forwarding a packet originating from a misbehaving node can drop the packet to encourage it to act sociably [14]. Ostracising the misbehaving node rewards it by lightening its burden. This rating method also penalizes properly behaving nodes by tasking them with more forwarding.

3.1.2 CONFIDANT

Buchegger and Le Boudec [7] present extensions to their CONFIDANT [6] reputation system that incorporates second-hand information in a controlled fashion. Their approach uses both first- and second-hand information. Nodes observe whether or not their next hop forwarded the packet successfully or not. Nodes compare their assessments of others with the second-hand information they received; the node will only use it if it does not deviate considerably from its own assement. The accepted second hand information only contributes slightly. These steps help to protect against

lying nodes and also help decrease the detection time of misbehaving nodes.

CONFIDANT uses trust ratings to increase the response to misbehaving node detection. If a node receives second-hand information from a node it trusts, then it incorporates the new values without first checking the deviation, which speeds up the detection time.

When a node classifies another node as misbehaving, it isolates the misbehaving node from the network. Misbehaving nodes have an opportunity to rejoin the network in the future if they start behaving sociably again. CONFIDANT focuses more on recent behavior and introduces the idea of rating degregation. Their rating function has a weight that affects how much past behavior influences the currernt rating. The fading mechanism gives misbehaving nodes an opportunity to redeem themselves.

The authors introduce some new attacks on reputation systems and detail how CONFIDANT performs with them. In a brainwashing attack, colluding nodes near to the victim publish false reports and trick the victim into believing them. CONFIDANT succumbs to this attack, but the effects are not permanent if the victim moves out of range from the attackers. Nodes performing an intoxication attack first gain the trust of the victim, then they start to inject false information. CONFIDANT protects againts intoxication with rating degregation and only admitting second-hand information that does not deviate significantly from its own values.

3.2 Virtual Currency

Some researchers have proposed using economic and game theoretic approaches to promote cooperation such as in [18]. In a market driven environment, consumers compensate service providers with currency. Intermediate nodes along a route provide service by forwarding packets. They do work using their own resources on behalf of the source node. The introduction of currency into the network creates new problems like where to store the currency, how to charge for services, and ensuring honest behavior. The following projects address these issues.

3.2.1 Nuglets

Buttyan and Hubaux [8] take an approach inspired by economics and use a virtual currency exchanged for packet forwarding. Each node in the network has a number of tokens called nuglets used for forwarding. They propose two variants of payment for forwarding. In one method the forwarding node pays the next hop along the route. The other method charges the next hop to forward the packet. The price depends on a number of factors including: current energy level, number of nuglets possessed, and the amount of energy required to transmit the packet.

In the Packet Purse Model, nodes pay others for forwarding packets. The source node calculates the expense associated with the route and deposits an suitable number of nuglets into the packet. Each of the intermediate nodes must pay the next hop, but each hop withdraws this cost from the packet to compensate their own expense. The authors argue this model helps deter senders from transmitting useless data, but they concede that computing the cost for the packet accurately is difficult.

Their other method tasks the next hop with paying for the packet. Nodes have incentive to forward packets because they will be able to earn nuglets from selling the packet to their next hop. This frees the sender of the burden of guessing the number of nuglets to load a packet with. Auctioning provides one possible means for determining the next hop. Naturally, this increases the overhead of the protocol since each hop along a route acts as an auctioneer. Neighboring nodes must exchange messages to bid on the packet at every hop; this also increases routing overhead considerably.

The virtual currency method proposed they propose rests on a number of assumptions. Each node has a security module responsible for storing, crediting, and deducting nuglets. This module also holds: the nodes public and private keys, the manufacturers certicate and public key, as well as session keys. The security module resides in tamper proof hardware asssumed to be made from a handful of of manufacturers. These assumptions limit the applicability of the approach to certain

environments.

3.2.2 SPRITE

Zhong et al. [37] introduce the SPRITE system, which is used to encourage cooperation in ad hoc networks by offering incentives to intermediate nodes acting as forwarders. In their system nodes receive credit for forwarding packets. SPRITE uses a centralized credit clearing house used to determine how much the sender must pay each of the intermediate nodes. It also incorporates a number of measures to prevent cheating in the system.

The SPRITE system consists of a collection of mobile nodes on a wide area wireless network connected to a credit clearing service over the Internet. SPRITE is based around the idea of paying intermediate nodes for their efforts. Instead of requiring tamper-proof hardware responsible of for credit manipulation, SPRITE uses receipts. Nodes send these receipts to the credit service when they reach a high-bandwidth connection and have backup power. Nodes may purchase credit through real-world funds, but the ideal method of income is forwarding packets for others.

SPRITE offers incentives to nodes to forward packets for others and to dutifully report their receipts. The authors chose the sender to credit the intermediate nodes for several reasons. Among these, attackers could drain destinations of credits by sending them a large number of packets. The credit clearing service gives credit to each of the nodes along the path that returned receipts. In the event of a packet drop by an intermediate node, the node before the faulty node also receives credit but not as much as those before it. The authors describe a situation where the source node and node before the faulty link can collude and increase their income, so their credit system places a lower credit value on the last node to prevent this possiblity. Misbehaving nodes not forwarding the packet do not receive credit.

This method relaxes some of the assumption made in [8], but it has not done away with all of them. Their system uses asymmetric public key cryptography and digital signatures, which require significant computation for resource constrained devices. It also presumes a connection to

the Internet, which may not be an option for all ad hoc networks.

3.3 Cryptographically Inspired Approaches

Many techniques have been proposed using cryptography for ad hoc network security [17, 34, 13]. Traditional wired network cryptographic techniques [26] do not translate to wired networks well [5]. Prime factoring computation used in many cryptography algorithms requires many costly CPU cycles. Potlapally et al. [32] support this claim with their study on energy usage of security protocols for portable devices. They show the high cost of asymmetric cryptography. Ad hoc networks do not rely on pre-existing infrastructure, so this makes using certificate agencies difficult. Cryptographic approaches for ad hoc routing tend to use light weight algorithms to avoid the cost for complex calculations.

3.3.1 Secure Routing Protocol

The Secure Routing Protcol (SRP) [27] uses security associations between source and destination nodes to protect the integrity of the route discovery process. SRP verifies the contents of route discovery packets with message authentication codes using the shared secret. This method benefits from not relying on pre-existing infrastructure like certificate authorities and does not require intermediate node computation. Communicating nodes create the message authentication codes with a shared secret obtained through methods like key exchange or loaded before deployment.

When a node initiates a route query, it chooses a random number called the query identifier. Along with this, the sender maintains a query sequence number for each of destination it communicates with. It places a message authentication code in the header based on the underlying protocol headers and the secret key. This is propagated in much the same was as in other reactive routing protocols, i.e. expanding ring propagation. Intermediate nodes extract the query identifier from the packet and only rebroadcast it if it has not seen it before. Intermediate nodes also keep track of the frequency it receives route queries from a given source. This prevents malicious nodes from flooding the network with queries. Once the request reaches the destionation, it calculates

the hash for the message authentication and creates a route reply packet.

This scheme works against many attacks launched by a single node, but does not ensure protection against colluding nodes. This method only secures the route discovery process, but does not protect the communication afterwards. This still leaves the packets vulnerable to tampering by malicious nodes and spurious packet drops by misbehaving nodes.

3.3.2 Ariadne

Ariadne [16] is a routing protocol geared for securing routing messages. It uses symmetric key cryptography to help deal with problems like compromised nodes and denial of service attacks. It has flexibility in that it provides three means of authentication: shared secrets between all nodes, a shared secret between communicating nodes using broadcasting, and digital signatures. In their paper, Perrig et al. focus on their TESLA [31] protocol, which allows them to optimize their protocol accordingly.

Ariadne uses hash chains to help safeguard the route request process. The initial packet contains fields similar to a DSR route request, but it also contains a message authentication code based on header information used to start the hash chain process. Intermediate nodes receiving the request first verify the validity of the request by comparing header information to their route request table. If the request is valid, then the node adds itself to the hop list and includes itself in the hash chain. The destination then waits to receive the keys, verifies the hash chain, then creates a reply.

Ariadne added security enhancements to the underlying routing protocol at a small cost. The authors tested Ariadne with the unoptimized version of DSR and showed an improvement in packet delivery at lower mobility, a slight increase in overhead, and lower latency. The optimized DSR variant outperformed Ariadne in these metrics, but it did not have the security provided by Ariadne.

3.4 Biologically Inspired Approach

Areas outside of computer science encounter similar problems in their respective fields. Modeling helps take a solution from the other domain and apply it to networking. The following work models

a network on biological principals.

3.4.1 Artificial Immune Systems

Le Boudec and Sarafijanovic [4] model ad hoc networks as an artificial human immune system capable of adapting to new threats. They base their work on the adaptive immune system, which has a system for creating detectors and only maintaining detectors that match a particular pathogen. They use the danger signal model, which is also based on response by cells, to increase the effectiveness of misbehavior detection.

Each part of the network corresponds to a component found in the immune system. The network consists of self cells and non-self cells, which are misbehaving nodes. This system observes DSR packet headers and treats them as antigens. It builds a list of behaviors witnessed over a fixed period for each of its neighbors. Antibodies act as detectors and try to match non-self antigens. If an antibody matches an antigen representing misbehavior in a fixed period of time, the node is considered of be suspicious. Problematic nodes become monitored at first and deemed misbehaving after the probability they are misbehaving exceeds a threshold.

Their system helped to detect misbehavior better with more self-cells, but it also required a learning phase where it assumes no misbehavior occurs. They concede the problems with this assumption and provide an solution to this in [35]. In response to this, Le Boudec and Sarafijanovic propose a virtual thymus that eliminates the need for the initial learning phase.

This work is interesting because it attacks the problem of misbehavior from an angle few have looked at before. The authors continue to refine their model and search for better detection mechanisms.

3.4.2 The Resurrecting Duckling

Stajano and Anderson [36] describe a security mechanism for wireless ad hoc environments drawing from wildlife biology. They focus on wireless devices where the user of the device varies over time and the usage depends on context. This is driven by real world ad hoc networking and brings

up interesting points for ad hoc security researchers to consider.

In the case where the user of a device might change with time, the devices need a mechanism to adapt to the new user and their permissions. They discuss a doctor connecting a wireless thermometer to their palmtop computer. When the doctor has imprinted the device, they will be able to use the device on patients. On the other hand, when a technician uses the device to calibrate it, their use is limited to the calibration equipment.

Stajano and Anderson propose the idea of secure transitive associations. When a device is activated, the user binds the device to themself until the user finishes with it. Instead of using cryptography for the initial secret exchange, they propose using physical contact between the the interacting devices. This initialization process mimics the way that ducklings identify their mother. In their system, the first person to send the device a secret key becomes the parent. When the user is done using the device, they are authorized to dismiss the device. The device then awaits the next user to come along and imprint it.

Tamper resistance helps prevent against malicious activities during the lifespan of the device. If an attacker is able to comprimise a node and extract the keys out of a device, they greatly reduce the security of that system. This property is difficult to guarantee and was looked at by [16] in terms of routing security.

3.5 Other

3.5.1 TWOACK

Balakrishnan et al.[3] introduces the TWOACK scheme for detecting failure to forward packets. TWOACK lets a node know whether or not the node two hops away has forwarded a packet. The authors claim that failure to receive such an acknowledgment indicates misbehavior on the part of the node two hops down the source route.

The TWOACK system is built on top of an arbitrary source routed protocol. Every time a node forwards a packet, it stores the packet id in a table until it has received a TWOACK packet.

The same node forwarding a packet emits a TWOACK packet that travels two hops prior to that node. Both of the nodes before the current forwarder compare the packet id contained in the acknowledgement and clear the packet from their table. If the previous two nodes do not receive such an acknowledgement, then the current forwarder is considered misbehaving.

TWOACK naturally increases the protocol overhead since every packet forward results in a packet that travels two hops back. It increases the overhead by 75%. TWOACK achieves a higher packet reception as the percentage of misbehaving nodes increases compared to an unaugmented copy of the routing algorithm, DSR in this case.

The authors propose the S-TWOACK scheme to help reduce the control packet overhead. S-TWOACK sends back a S-TWOACK packet after receiving a certain number of packets. This decreases the overhead, but the authors say this makes the problem of false-alarms more noticeable. Their simulation results showed the S-TWOACK scheme generating less overhead than both DSR and TWOACK.

3.5.2 Adaptive Path Selection and Loading

The Adaptive Path Selection and Loading(APSL)[22] method uses multiple node-disjoint routes with Reed-Solomon(RS) codes to deliver more packets with node misbehavior and naturally caused packet loss. The destination can use the RS codes to reconstruct lost data under certain conditions. If it receives k symbols it can build the data without receiving the lost packets. Kefayati et al. introduce path state information to judge the quality of the routes. They measure quality in terms of path stability and availability.

APSL has a closed-loop feedback mechanism to help select the best routes. The destination calculates the path state information and then sends this to the source. The destination can send this information on an ACK packet. The source also computes path state information implicitly from data it receives from the reverse of that path.

APSL maintains a high packet delivery ratio even at high percent of misbehaving nodes. The

overhead also remained roughly constant as the percentage increased. The algorithm they compare their solution to experienced a marked decrease of packets.

CHAPTER FOUR

HISTORY-BASED ROUTE SELECTION

This chapter introduces history-based route selection. We describe the structures and algorithms necessary for our method to work. Finally, we conclude with an example illustrating the workings of history-based route selection in a sample network.

4.0.3 Overview

History-based route selection augments the underlying DSR protocol with the ability to maintain past performance of nodes and choose routes based on their quality. The quality of forwarders composing a route affects the performance of that route. Nodes prone to dropping packets detract from the overall utility of a path. Without maintaining any sort of history, a route selection algorithm cannot differentiate between good and bad nodes. The original DSR protocol selects routes soley on the length of the path and has no recollection of whether nodes along that path have performed well in the past. It does not associate any form of quality with the nodes or routes. This can lead to performance degragation as the number of misbehaving nodes increases as shown in chapter 5.

The quality of a node depicts the overall reliability of that node. Unreliable nodes consistently drop packets while a high quality node forwards packets regularly and does not often drop packets. Nodes with a better track record of dutifully forwarding packets have more utility than nodes with a poor history. In order to evaluate the goodness or badness of a node, there must be a means to evaluate their performance.

Our system uses packet counts to represent the performance of other nodes. Packet counts represent forwarder behavior well: a forwarder either forwarded the packet or it did not. These values are collected at no charge from the underlying routing layer. When an intermediate node forwards a packet for the sender, it demonstrates its good citizenship. The cooperating node receives an increment in its good forward count. Unacknowledged protocols like UDP do not have a feedback

Table 4.1: History-Table Format

Node Id	Good	Bad
	⋮	
ith node	#	#
	⋮	

mechanism to alert the sender of a successful receipt by the destination, but other information available to the host gives negative feedback, i.e. the packet did not reach the destination. In DSR, route error packets arise when this happens. Receipt of error packets add to the negative packet count for the node responsible for the error.

These changes do not require heavy modification of the underlying routing layer. History-based route selection only uses routing layer data, so it supports the idea of modularity. It does not rely on a particular transport layer above it, nor does it require a specific link layer protocol. This system is suited for real world operation due to its simplicity and ease of implementation.

4.0.4 History Table

History-based routing maintains a structure called a history-table to record its experiences with other nodes. When a node joins the network, it has an empty history table and empty route cache. The table entries are indexed with the node id and have two fields: the number of good and bad forwards. A good forward represents the case where node successfully forwarded the packet to the next hop. A bad forward indicates the node dropped the packet. These two values constitute 8 bytes of storage per node, which does not place a heavy burden on the nodes memory.

The history table grows as the node interacts with the network and gains routes. Nodes gain routes directly from route replies and secondarily from overheard packets. When it gains a new route, it adds the routes to its cache. New nodes receive an entry in the history-table and have an initial value of zero for both the number of good and bad forwards. Over time, the underlying routing layer removes routes from the cache, but the history table still holds the statistics for each of the nodes along the routes. Routes are removed for several reasons. When the route cache

becomes full, the underlying protocol removes older, stale routes to make room for newer routes. The receipt of route errors also forces out all cached routes including the erroneous node. Nodes do not have to be recipient of route error packets; overhearing or forwarding a route error can lead to route removals as well.

4.0.5 Rating Scheme

The route selection algorithm uses the history-table and the route cache to select the route with the best average rating. Each node receives a rating based algorithm 1. This gives a rating between 0.0

Algorithm 1 Node Rating Function

```
if ∃ node n then
    return good forwards/total forwards
else
    return 1.0
end if
```

and 1.0 where 1.0 represents a 100% delivery record. In the event the node just entered the table and has no past interaction, then it is assumed to be properly functiong until proven wrong. This is a simple rating scheme based on algebra, so it does not require intensive computation nor does it require much storage. Given these values, when the source requests a route from its cache, the cache performs the algorithm 2.

Once the route with the best rating has been selected, the sender credits each of the nodes along the route by incrementing the good forward count by one. Since this scheme works with an unacknowledged protocol, it does not know if the packet has arrived successfully at the destination. This scheme takes a innocent-until-proven-guilty approach and assumes the packet does indeed reach the destination unless it hears otherwise.

The source node penalizes intermediate nodes upon the receipt of packets containing route errors. For each of the route errors contained in the packet, the sender determines the source of the error and increments the bad forward table entry to reflect the misbehavior. Furthermore, since the sender assumed correct behavior when sending the packet along the route with the node, it also

Algorithm 2 Route Selection Algorithm

if $\exists$ routes R to destination d **then**
  $biggest = 0.0$
  for all routes $r \in R$ **do**
    H $\leftarrow h_1, h_2, \ldots, h_n$ where h_i is the ith hop of r
    rating $\leftarrow \frac{\sum_{i=1}^{|H|} rating(h_i)}{|H|}$
    if rating $>=$ biggest **then**
      biggest $\leftarrow$ rating
    end if
  end for
  return $r_{biggest}$
else
  return *null*
end if

decrements the previously awarded good count.

4.0.6 Example

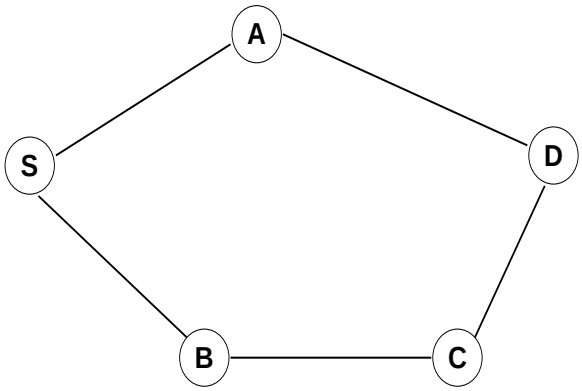

Figure 4.1: Example Network Topology

Consider the network shown in Figure 4.1. In this scenario, node S wishes to establish a connection with node D. It has the values in its history table shown in table 4.2 and routes to node D as shown in 4.3. It has not previously interacted with node B, so it has a rating of 1.0. Routes (1) and (2) have an average intermediate node rating of .95, but route (1) is the shortest so node S uses this route.

Next, suppose that node A drops a packet. S removes route (1) and any other routes with node

Table 4.2: S's History Table

	Good Forwards	Bad Forwards	Rating
A	114	6	.95
B	0	0	1.0
C	9	1	.9
	⋮	⋮	⋮

Table 4.3: S's Routes to Node D

Route	Rating
(1) S → A → D	.95
(2) S → B → C → D	.95

A. It has an alternative route to node D, so it sends subsequent packets to D along route (2). As it sends packets along this route, the intermediate nodes receive credit in the good forward count.

S continues to send packets along this route until C drops a packet and it receives a route error from node B. After receiving the route error packet it removes route (2) from the cache and has no routes to D. It must initiate a route request to find a new route. Both node A and C are working correctly, so they return the routes $S \rightarrow A \rightarrow D$ and $S \rightarrow B \rightarrow C \rightarrow D$, respectively. Node S bases its decision on 4.4. Route (1) has a rating of .942 and route (2) has a rating of .967. Since route (2) has a better average node rating, S chooses this route.

Table 4.4: S's history table after more time

	Good Forwards	Bad Forwards	Rating
A	114	7	.942
B	20	0	100
C	28	2	.93
	⋮	⋮	⋮

CHAPTER FIVE

PERFORMANCE EVALUATION

This chapter addresses the simulation efforts used to test the effectiveness of history-based route selection. We detail the parameters used for the simulation including the environment setup, mobility model, and traffic patterns. We describe and justify the metrics used to measure the performance. Finally, this chapter presents the simulations and provides insight into the experimental results.

5.0.7 Simulation Setup

There were two major thrusts in the simulation design. One set of experiments varied the percentage of misbehaving nodes from 0% to 40% at intervals of 5% to test how our algorithm performed with varying degrees of misbehavior in the network. The other major set of experiments varied the time to investigate if more information leads to better route selection. Nodes in history-based DSR gather more information about the network as time progresses, so they will be able to make better informed choices about route selection.

We conducted our simulations using the ns2 [12] simulator. This is a discrete event simulator widely used in networking research. Our simulations use the Monarch extensions for simulation wireless mobile devices. The simulations occured in an 1000x1000 meter area with 100 nodes. Nodes started at randomly determined (x,y) coordinates assigned through a TCL script. Node mobility followed the random waypoint model [9]. Nodes move with constant velocities in a line under the random waypoint model. The node rests at the destination for the duration of the pause time. It randomly picks both a direction to travel in and a constant velocity to use. The maximum movement speed bounds the random value. We used the parameters shown in table 5.1. 20 m/s corresponds to the movement speed of an automobile while 5 m/s is a bit faster than human walking speed. This mobility model acts as a worst case scenario because the nodes do not move in any predictable fashion. The random waypoint model creates interesting topology changes. This behavior creates both expected and unexpected behavior that help to exercise the protocols.

Table 5.1: Random Waypoint Model Parameters

Maximum Speed (m/s)	Pause Time (s)	Relative Mobility
5	30	Lowest mobility
5	5	Slow movement, more mobility
20	30	Fast movement, long pause
20	5	High mobility

Each set of experiments also used either 10 or 15 connections of CBR traffic at several rates detailed in the following subsections. A TCL script randomly chose source and destination pairs. Both source and destination nodes could belong to multiple connections. Nodes flagged as misbehaving were also allowed to participate in connections. All connections started at time 0 and data gathering started at 60s. This allowed routes to form and traffic to stabilize. These are reasonable conditions similar to those used by others and permit comparison.

The misbehaving node behavior varied with the experiment goals. The misbehaving nodes all started at the start of the simulation. They dropped all packets except for those containing route replies and requests in order to place themselves along routes. The nodes were randomly chosen and placed throughout the network. The experiments concerning time used 15% misbehaving nodes. This number of misbehaving nodes tested the algorithm while not completely degrading the network.

Our discussion on the results focus on the scenarios with the most challenging conditions, because they reflect performance in the worst-case scenario. Higher mobility scenarios present more challenges to protocols because of their changing topology.

5.0.8 Metrics

We evaluated history-based route selection using two sets of related metrics: general network performance and stability. Metrics like throughput and delay reflect the effectiveness of the network as a whole. A healthy network delivers packets in a timely fashion with a reasonable number of packet drops. Out of these measures, we particulary looked at the number of data packets received

over the course of the simulation to represent the throughput. The time the packet left the sender and arrived at the destination measured the delay. Connection sources of both the length-based selection method and history-based route selection method used a constant bit rate stream of UDP packets at 12.2 kbps to send data packets. A higher count depicts more ability to receive data and reflects the overall utility of the network.

Comparing the type and number of control packets shows the effects of the algorithm on network stability. A stable routing algorithm has less overhead because the routes it chooses are reliable. In this same vein, stable routes remain effective longer and produce fewer route errors. The percentage of control packets of the overall traffic is also important. A higher percentage indicates the nodes must spend time processing protocol information rather than sending data packets on their way. To this end, we recorded and counted the number of route requests, route replies, and route errors. We measure the overhead using the number of packets received rather than sent because it better depicts the effort of intermediate nodes receiving route requests better than only using sent packets. Counting the a route request sent by a source only shows effort on the sender's behalf and does not depict work done by others. These packets are flooded throughout the network, so intermediate nodes most expend their resources processing these packets and acting upon them.

5.0.9 Varying Degrees of Misbehavior

Progressively increasing the number of misbehaving nodes tested the performance of history-based route selection under a more hostile environment. As the percentage of misbehaving nodes grows, the chance to have a misbehaving node along a route increases. Figures 5.1 through 5.5 represent simulation setup using 20 m/s maximum speed and a 5 second pause. This high mobility scenario changes the topology of the network more often than slower runs, which helps test the adaptibility of the algorithms.

Our results show the history-based selection scheme achieving higher aggregate packet reception and less control packets. Both routing methods decreased their delivery effectiveness at

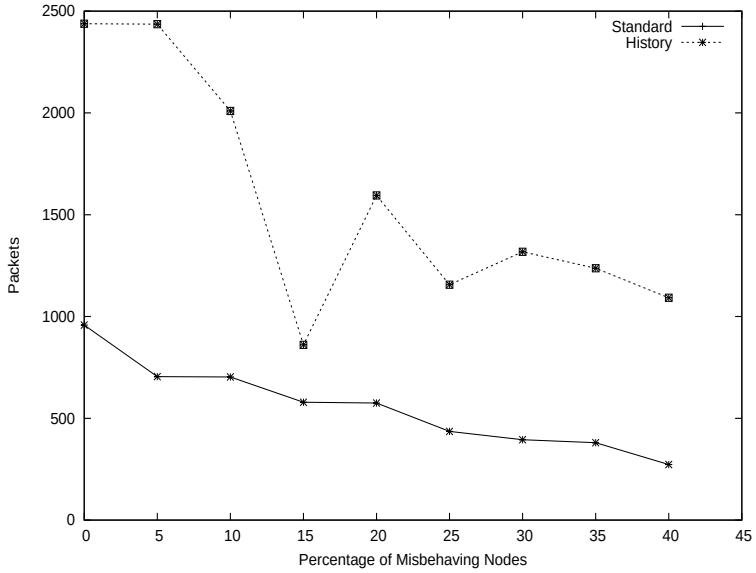

Figure 5.1: High Mobility: Data Reception

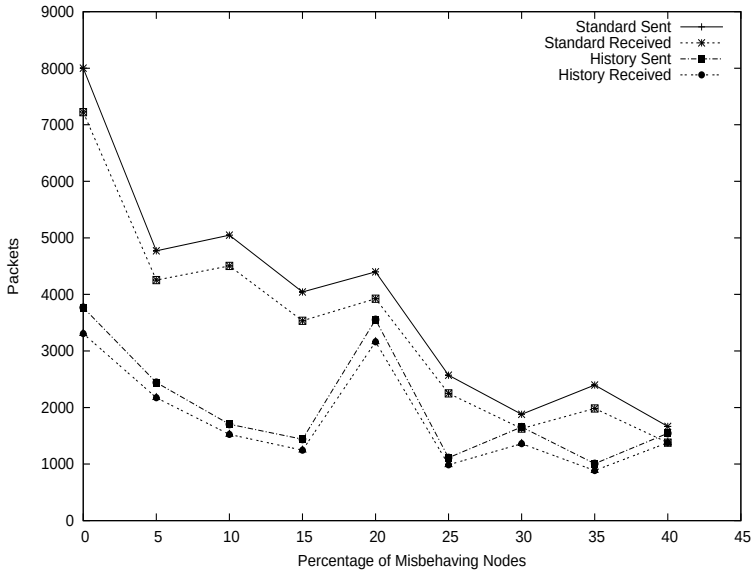

Figure 5.2: High Mobility: Control Packets

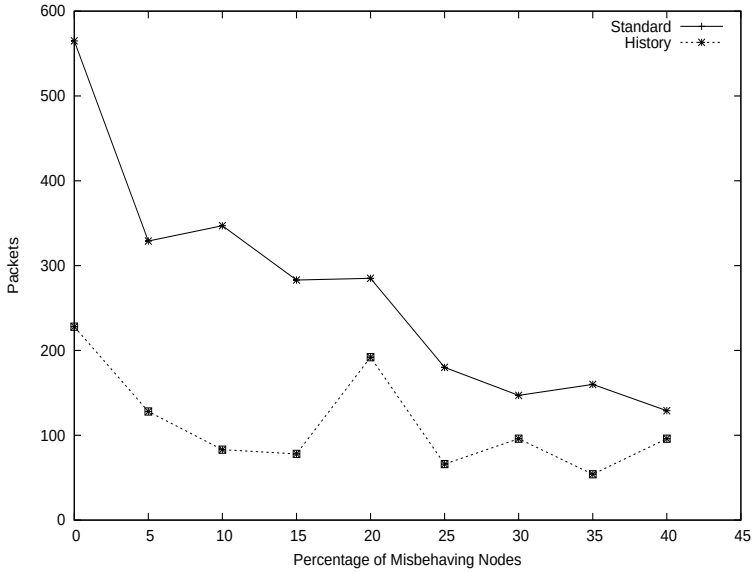

Figure 5.3: High Mobility: Route Replies

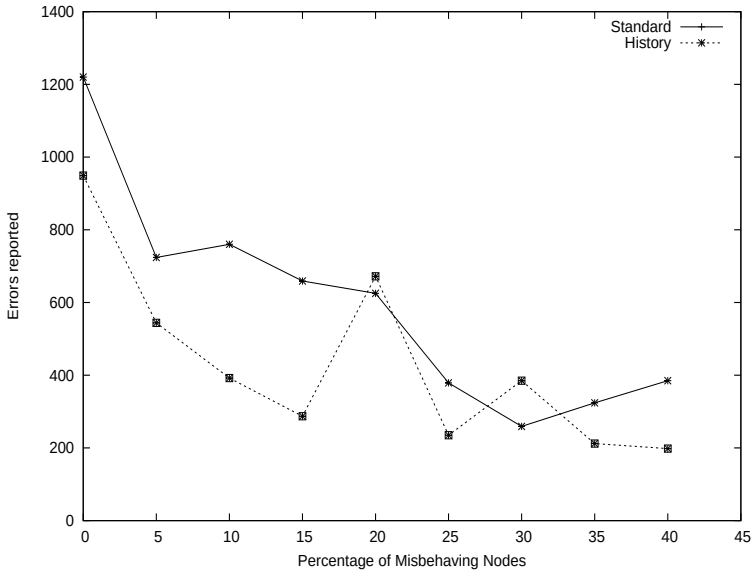

Figure 5.4: High Mobility: Route Errors

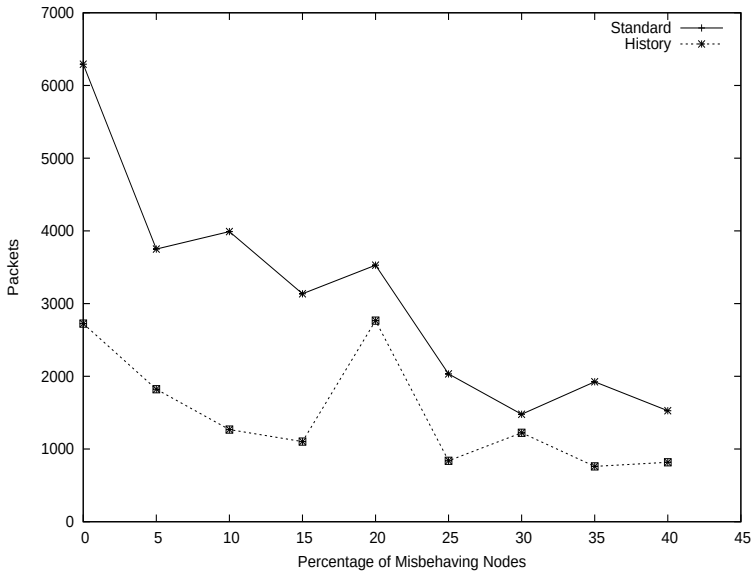

Figure 5.5: High Mobility: Route Requests

roughly the same rate, but the history-based scheme had a markedly higher packet delivery than the standard counterpart. History-based routing initiated fewer route queries. This indicates the routes it chose were more stable. Length-based selection also experienced more route errors than history-based routing. This indicates that it chose routes with nodes more prone to drop packets.

History-based route selection and its length-based counterpart performed comparably at low mobility. Figure 5.6 shows both algorithms saw similar levels of packet reception as the percentage of misbehaving nodes increased. Low mobility does not expose nodes to other nodes as much as with higher mobility. Sources that have come into contact with more nodes have a greater range of forwarders to choose from. Both selection methods generated an equivalent number of control packets and packet drops as supported in figure 5.7 and 5.8.

Average Node Rating vs. Minimum Node Rating

We performed the percentage variation simulations using the average node value and compared this with using the minimum node value to determine the routes utility. The purpose of these experiments was to determine how much using the minimum node value along a route would alter

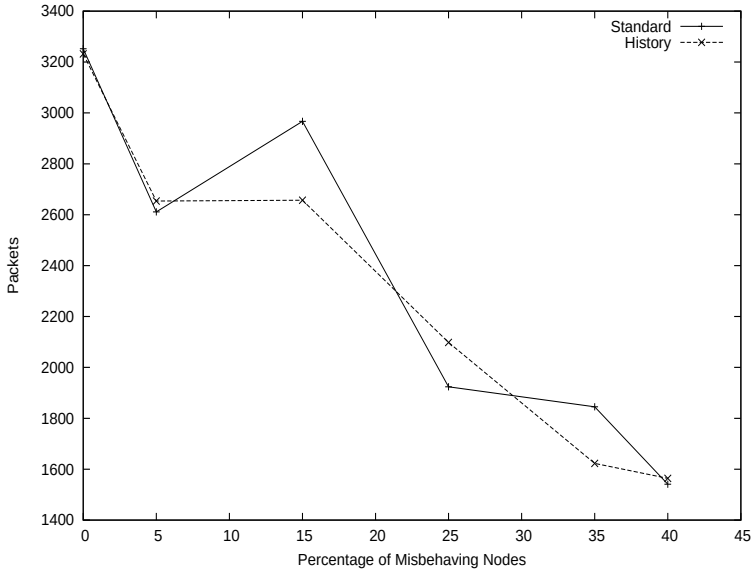

Figure 5.6: Low Mobility: Data Reception

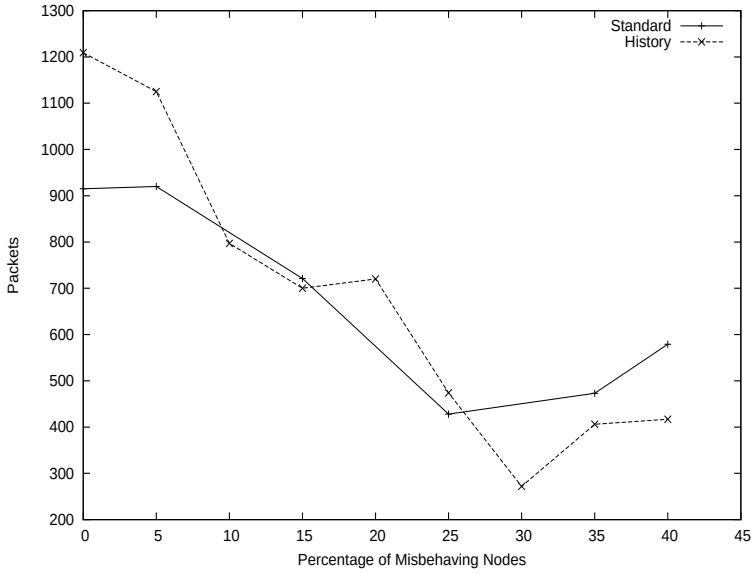

Figure 5.7: Low Mobility: Route Control Packet

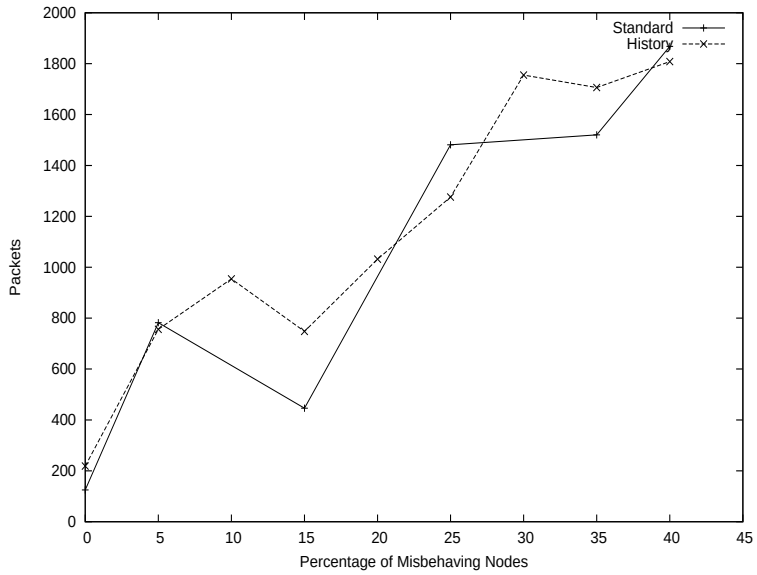

Figure 5.8: Low Mobility: Packet Drops

the selection algorithm performance.

The experimental setup followed the previously mentioned percentage experiments. We used a high mobility scenario with a maximum speed of 20 m/s and a pause time of 5 seconds. The simulations lasted four minutes with an initial 60 second period to allow the network to stabilize. The collected data represents three minutes of traffic following this period.

The results indicate that the choice between the minimum node value and the average node rating does not significantly affect the performance of the algorithm. Figures 5.9, 5.10, and 5.11 support this consistently. This similarity does not change as the percentage of misbehaving nodes increases.

5.0.10 Sensitivity to Time

The time sensitivity experiments used a simulation setup similar to the percentage experiments. The nodes were randomly distributed in a 1000x1000 meter area. Mobility followed the random waypoint model and used the same parameters described in table 5.1. Connections were determined randomly according to the previous experimental setup. The percentage trials used four

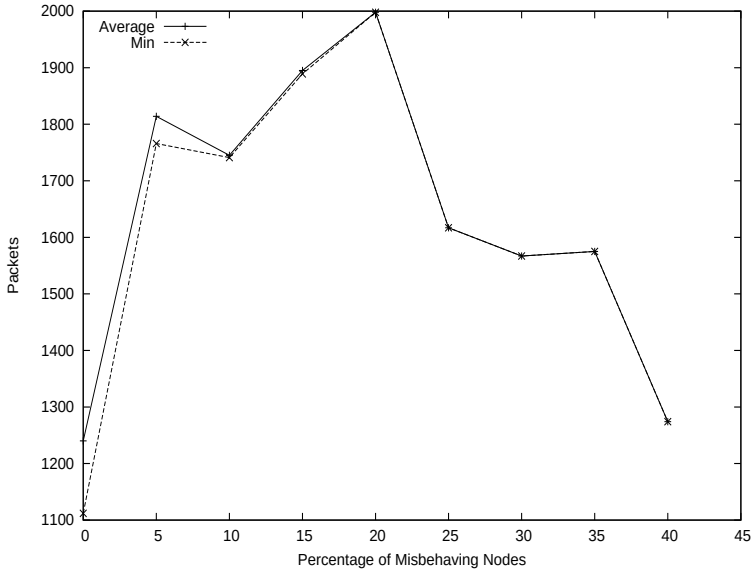

Figure 5.9: Min vs. Avg.: Data Packet Reception

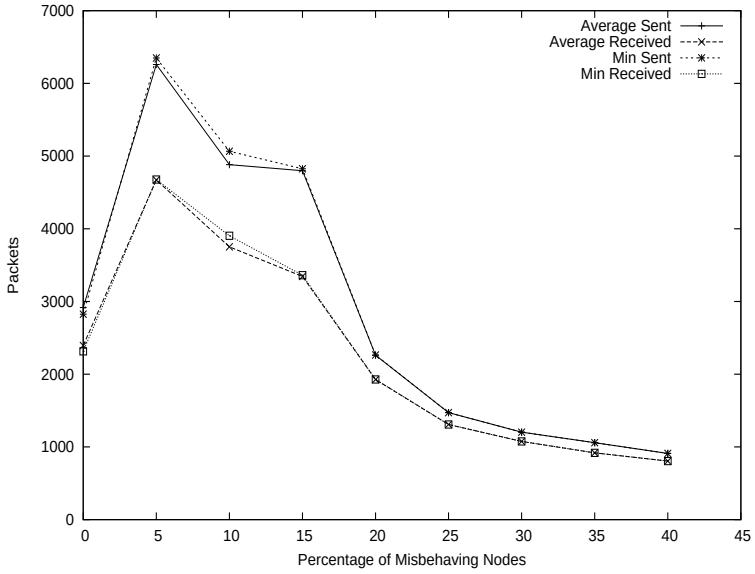

Figure 5.10: Min vs. Avg.: Control Packets

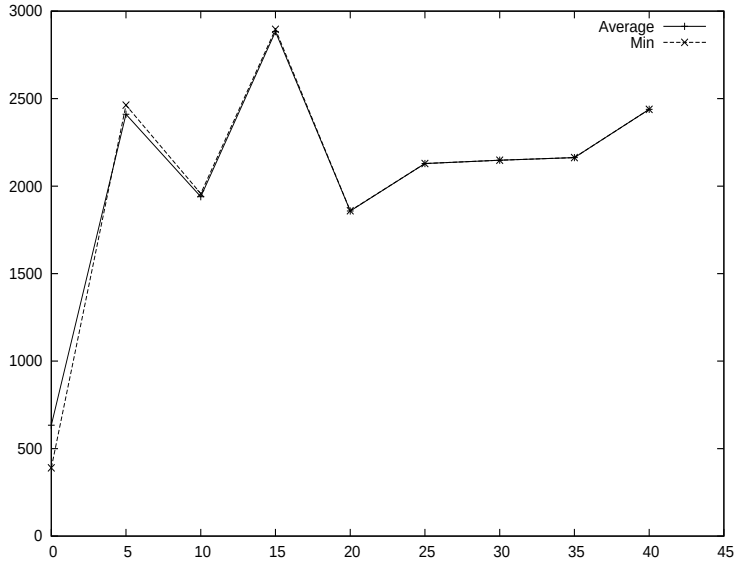

Figure 5.11: Min vs. Avg.: Packet Drops

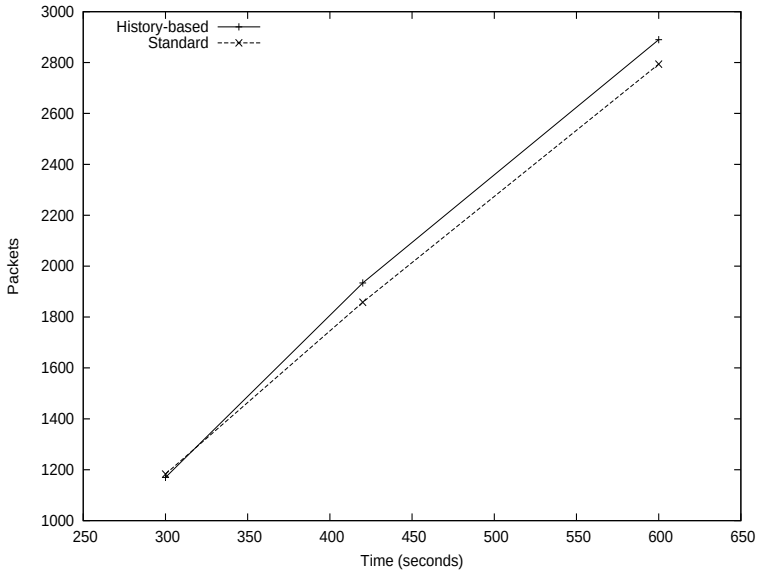

Figure 5.12: Time Sensitivity: Data Reception

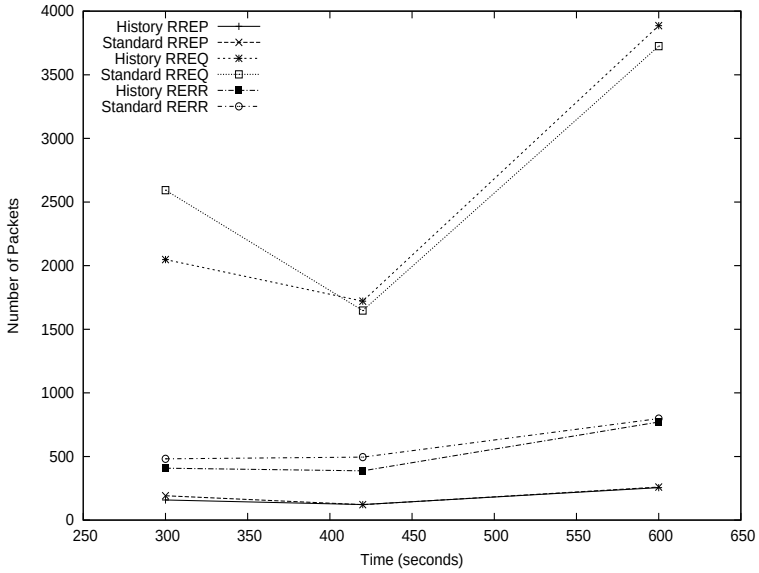

Figure 5.13: Time Sensitivity: Overhead

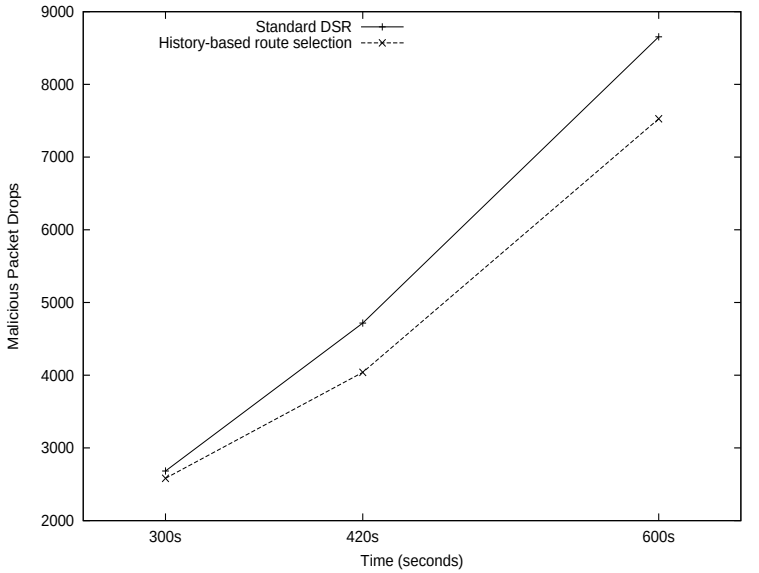

Figure 5.14: Time Sensitivity: Packet Drops

minutes time, so we used longer time periods. The maximum time of the simulation was limited by the hardware of the machine running the simulations. The time sensitivity trials used 15% misbehaving nodes. This percentage exercises the algorithms' performance due to misbehavior, but it is also allows the network to function.

History-based routing collects evidence of node performance over time, so it gains a better picture of the network with more time. Figure 5.14 shows performance improvement with longer simulation times. Initially both routing algorithms behave similarly at 300 seconds. History-based route selection starts to achieve higher aggregate data reception with time. The data points on Figure 5.14 illustrate the difference between the two algorithms increases with longer simulations. At 15% misbehavior, both of the selection algorithms had comparable routing overhead. As the time increased, history-based selection avoided more packet drops than standard route selection. The history-dependent scheme chose higher quality routes as evidenced by the figures.

5.1 Discussion

At lower movement speeds, both algorithms behaved comparably in terms of both general network performance and stability. Figure 5.6 shows no significant difference in the number of data packets received. Figure 5.7 and 5.8 show comparable overhead and packet drops respectively. In these conditions nodes remain in place more and do not move out of transmission range from their neighbors as often than a higher mobility scenario. The simple length-based approach picks short routes to a destination; if the route breaks it can quickly choose another short route. Length-based route selection performs well with lower mobility, but it starts to experience lessened performance as the mobility increases.

Higher mobility resulted in history-based routing performing better than the standard algorithm. The scenario with a maximum speed of 20 m/s and 5 second pause time exercised the algorithms the most, but the 5 m/s and 5 second pause time case also created considerable mobility. The more mobile nodes move out of transmission range of their neighbors, more links

break and result in route error generation. The changing topology places more of demand on the route selection algorithms since routes change more often. The introduction of misbehaving nodes complicates the already challenging network condition.

History-based route selection experienced greater aggregate network performance even with no misbehaving nodes. Length-based route selection received nearly 1000 packets while history-based route selection received close to 2500 data packets. At the same percentage, length-based selection sent more overhead packets then history-based selection. Even without misbehaving nodes, the history information helped to select better routes to destinations. A highly mobile node lying along a short route seems appealing using route shortness as a metric; however, if past routes have included the highly mobile node only to break later due to mobility, then the history-based selection method captures this behavior.

The more interaction with nodes helps to make better route selections. Scenarios with longer durations showed an improvement in performance as time went on. The longer time allowed nodes to collect more packet counts and build their history-table. Results of trials with higher packet rates support this claim. Previous work with varying the packet rate showed improvements in performance at higher rates. Our mechanism works with packet counts, so the more packets observed by the sender help to build their view of the network.

The lower number of control packets received demonstrated the stability of our algorithm. Routes that last longer have utility to both the sender and the network. The sender does not need to flood route request packets as much, which saves the network from propagating the requests. Fewer requests translates to fewer replies; this further reduces network control packet numbers. Stability also means fewer broken routes. Lower number of route errors helps the network from experiencing flooding associated with the query process. Our results show less overhead and higher data reception, which means that the nodes have spent more time transmitting data rather than control packets.

The number of control packets received decreased as the number of misbehaving nodes increased. The results show that the number of sent control packets decreased as well. In the worst case shown, 40% misbehaving nodes, control packets sent out had a high chance of being dropped by an intermediate misbehaving node. These control packets are important because they keep nodes updated about the status of the network topology. If a node does receive a route error packet, then it will continue to try and use the same route. If the route is defunct, then the packet drops will continue to increase.

CHAPTER SIX

SUMMARY

Ad hoc networks enable exciting new applications not possible with traditional fixed infrastructure wired networks. The changing topologies inherent in ad hoc networks, compounded with the lossy wireless medium, create a much different environment. The resource constrained mobile devices may opt to selfishly drop packets and preserve their own resources. Misbehaving nodes decrease the overall performance of the network considerably. These factors challenge protocol designers to create routing protocols capable of adapting to these factors.

In this paper we have described a light-weight route selection algorithm augmenting existing reactive routing protocols. It extracts and collects statistics from the underlying routing layer at no extra cost. History-based route selection does not require promiscuous listening or rely on the possibly corrupt opinions of other nodes. Using this information, it selects routes based on the quality of the nodes based on their performance history.

This simple scheme helped improve network performance at higher mobility scenarios. Higher mobility changes the network topology frequently and pushes the performance routing protocols. Our simulations revealed that history-based route selection performed well in not only in a high-mobility environment, but a high-mobility environment coupled with adversarial nodes. The network was able to successfully send more data than just using the underlying routing protocol. The network using history-based route selection had fewer routing control packets. Stable routes help show the quality of the algorithm. Longer last routes reduce the need for route inquiry.

This performance and ease of implementation suit real world implementation with some further modification. The history-table grows as time increases, but due to the limited memory of portable devices, the table must face an upper bound on size. One way to solve this is to introduce entry freshness. Each entry in the history table will have a freshness rating marking the last interaction with the node. This allows the system to purge entries that have become stale. When the table is

full, the node can remove the oldest entry in favor of the new entry.

Future work aims to improve route quality at the recipient side of a route request instead of just at the source. Receivers normally reply to the first request they receive as it assumes this comes from the shortest route. History-aware route replies create route replies coming from reliable routes. Instead of replying to the first request, history-based replies waits for a short period of time then chooses the best route among those it received. This helps tackle the problem of route selection at both ends of communication.

BIBLIOGRAPHY

[1] Opportunistic networking: data forwarding in disconnected mobile ad hoc networks. *IEEE Communications Magazine*, pages 134–141, 2006.

[2] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. A survey on sensor networks, 2002.

[3] Kashyap Balakrishnan, Jing Deng, and Pramod Varshney. TWOACK: preventing selfishness in mobile ad hoc networks. In *IEEE Wireless Communications & Networking Conference*, New Orleans, March 2005.

[4] J. Le Boudec and S. Sarafijanovic. An artificial immune system approach to misbehavior detection in mobile ad-hoc networks, 2003.

[5] Michael Brown, Donny Cheung, Darrel Hankerson, Julio Lopez Hernandez, Micheal Kirkup, and Alfred Menezes. Pgp in constrained wireless devices. In *9th USENIX Security Symposium*, pages 247–261, 2000.

[6] Sonja Buchegger and Jean-Yves Le Boudec. Performance analysis of the CONFIDANT protocol: Cooperation of nodes — fairness in dynamic ad-hoc networks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing (MobiHOC)*, Lausanne, CH, June 2002. IEEE.

[7] Sonja Buchegger and Jean-Yves Le Boudec. Self-Policing Mobile Ad-Hoc Networks by Reputation. *IEEE Communication Magazine*, 43(7):101, 2005.

[8] Levente Buttyan and Jean-Pierre Hubaux. Nuglets: a Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical report, 2001.

[9] T. Camp, J. Boleng, and V. Davies. A survey of mobility models for ad hoc network research. *Wireless Communications and Mobile Computing (WCMC): Special issue on Mobile Ad Hoc Networking: Research, Trends and Applications*, 2(5):483–502, 2002.

[10] David Culler, Deborah Estrin, and Mani Srivastava. Guest editors' introduction: Overview of sensor networks. *Computer*, 37(8):41–49, 2004.

[11] Prashant Dewan, Partha Dasgupta, and Amiya Bhattacharya. On using reputations in ad hoc networks to counter malicious nodes. In *ICPADS '04: Proceedings of the Parallel and Distributed Systems, Tenth International Conference on (ICPADS'04)*, page 665, Washington, DC, USA, 2004. IEEE Computer Society.

[12] Kevin Fall and Kannan Varadhan. *The ns Manual (formerly ns Notes and Documentation)*. http://www.isi.edu/nsnam/ns/doc/index.html.

[13] Manel Guerrero Zapata and N. Asokan. Securing Ad hoc Routing Protocols. In *Proceedings of the 2002 ACM Workshop on Wireless Security (WiSe 2002)*, pages 1–10, September 2002.

[14] Q. He, D. Wu, and P. Khosla. Sori: A secure and objective reputationbased incentive scheme for ad-hoc networks.

[15] Yih-Chun Hu and Adrian Perrig. A survey of secure wireless ad hoc routing. *IEEE Security and Privacy*, 2(3):28–39, 2004.

[16] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. In *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking*(MobiCom 2002), September 2002.

[17] J. Hubaux, L. Buttyan, and S. Capkun. The quest for security in mobile ad hoc networks.

[18] M. Jakobsson, J. Hubaux, and L. Buttyan. A micropayment scheme encouraging collaboration in multi-hop cellular networks, 2003.

[19] D. Johnson, D. Maltz, and J. Broch. *DSR The Dynamic Source Routing Protocol for Multihop Wireless Ad Hoc Networks*, chapter 5, pages 139–172. Addison-Wesley, 2001.

[20] David B. Johnson, David A. Maltz, and Yih-Chun Hu. The dynamic source routing protocol for mobile ad hoc networks (dsr), 2003.

[21] Chris Karlof and David Wagner. Secure routing in wireless sensor networks: Attacks and countermeasures. *Elsevier's AdHoc Networks Journal, Special Issue on Sensor Network Applications and Protocols*, 1(2–3):293–315, September 2003.

[22] Mahdi Kefayati, Hamid R. Rabiee, S. Ghassem Miremadi, and Ahmad Khonsari. Misbehavior resilient multi-path data transmission in mobile ad-hoc networks. In *SASN '06: Proceedings of the fourth ACM workshop on Security of ad hoc and sensor networks*, pages 91–100, New York, NY, USA, 2006. ACM Press.

[23] Leszek Lilien, Z. Huma Kamal, and Ajay Gupta. Opportunistic networks: Challenges in specializing the p2p paradigm. *dexa*, 0:722–726, 2006.

[24] Leszek Lilien, Zille Huma Kamal, Vijay Bhuse, and Ajay Gupta. Opportunistic networks: The concept and research challenges in privacy and security.

[25] Sergio Marti, T. J. Giuli, Kevin Lai, and Mary Baker. Mitigating routing misbehavior in mobile ad hoc networks. In *Mobile Computing and Networking*, pages 255–265, 2000.

[26] Raymond Panko. *Corporate Computer and Network Security*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2003.

[27] P. Papadimitratos and Z.J. Haas. Secure routing for mobile ad hoc networks.

[28] C. Perkins. Ad hoc on demand distance vector (aodv) routing, 1997.

[29] Charles Perkins and Pravin Bhagwat. Highly dynamic destination-sequenced distance-vector routing (DSDV) for mobile computers. In *ACM SIGCOMM'94 Conference on Communications Architectures, Protocols and Applications*, pages 234–244, 1994.

[30] Charles E. Perkins, Elizabeth M. Belding-Royer, and Ian D. Chakeres. Ad hoc on-demand distance vector (aodv) routing, 2003.

[31] A. Perrig, R. Canetti, D. Tygar, and D. Song. The tesla broadcast authentication protocol, 2002.

[32] Nachiketh R. Potlapally, Srivaths Ravi, Anand Raghunathan, and Niraj K. Jha. A study of the energy consumption characteristics of cryptographic algorithms and security protocols. *IEEE Transactions on Mobile Computing*, 05(2):128–143, 2006.

[33] Tim Roughgarden and Ea Tardos. How bad is selfish routing? *J. ACM*, 49(2):236–259, 2002.

[34] K. Sanzgiri, B. Dahill, B. Levine, and E. Belding-Royer. A secure routing protocol for ad hoc networks, 2002.

[35] Slavisa Sarafijanovic and et al. An artificial immune system for misbehavior detection in mobile ad hoc networks with both innate, adaptive subsystems and with danger signal.

[36] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. pages 172–194.

[37] Sheng Zhong, Jiang Chen, and Yang Richard Yang. Sprite: A simple, cheat-proof, credit-based system for mobile ad-hoc networks. In *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1987–1997, 2003.