Afora: Ad Hoc Routing in the Face of Misbehaving Nodes

by

Hu Imm Lee

B.A., Computer Science
Mount Holyoke College, 1998

Submitted
to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2002

Author .....
Department of Electrical Engineering and Computer Science
August, 2002

Certified by .....
Robert Morris
Assistant Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Students

Afora: Ad Hoc Routing in the Face of Misbehaving Nodes

by

Hu Imm Lee

Submitted to the Department of Electrical Engineering and Computer Science
on August, 2002, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

This thesis describes Afora, a multi-hop wireless ad hoc routing algorithm that finds working routes despite misbehaving participants. If one or more routes from a source to a destination exist that are entirely out of radio range of any attacker, Afora will eventually find and use one of them; this is the best that a routing protocol can do. This property holds even if misbehaving nodes generate or forge routing information, and even if they lie on the shortest path between source and destination.

Afora does not directly use or require any public key or certificate infrastructure. Instead, it assumes that applications (or higher-level protocols such as IPSec) verify whether they are talking to the desired peer, and that they notify Afora when this is (or is not) the case. Afora uses these notifications to decide whether to keep trying alternate routes until it finds one that works. Afora's key contribution is its ability to find alternate routes in the face of misbehaving nodes that actively generate false routing information.

Thesis Supervisor: Robert Morris
Title: Assistant Professor

Acknowledgments

This thesis is the result of joint efforts with my advisor Robert Morris and Enoch Peserico. Their inputs and contributions were valuable to this project. I would also like to thank the wonderful people at PDOS for their support and encouragements during these last two years.

Last but not least, heartfelt thanks goes out to Anjing for letting me use you as a punching bag during desperate times. Aishiteru yo!

For Mummy and Daddy

Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Current ad hoc routing protocols are vulnerable to malicious participants. Such participants could drop or corrupt packets instead of forwarding them properly, jam the transmissions of nearby nodes, or generate incorrect or forged routing protocol packets. The best a routing protocol can do in the face of malicious nodes is find a route that consists of benign nodes and is out of radio range (i.e. jamming range) of all malicious nodes. This thesis proposes Afora, a new multi-hop ad hoc wireless routing algorithm that is guaranteed to find such a route, if it exists; it can do this despite the kinds of misbehavior listed above.

Afora's basic approach is to keep trying different routes until it finds a *safe route*, one that delivers packets correctly. Afora faces two problems. First, it must be able to find all routes in the face of incorrect routing information generated by malicious nodes. Second, it must be able to determine whether it has found a safe route.

An Afora source looking for a route to a target floods a query over the whole network, in a manner similar to Dynamic Source Routing [1, 10]. These queries do not accumulate source routes; their only purpose is to reliably reach the target. The target floods a reply back to the source on receipt of a query, and the replies accumulate source routes. A probabilistic choice by each node of which reply to forward ensures that the source sees a random selection of routes in response to each query, even if adversaries are generating incorrect routing packets. A source sends repeated queries to discover different routes,

until one of them is found to deliver data packets correctly. The key property of Afora's probabilistic route selection is that if a safe route exists, Afora is guaranteed to eventually find it.

Afora does not itself attempt to determine whether a route it has found is safe. Instead, it assumes that some higher level software will tell it whether packets are arriving successfully at the intended destination. In a simple system, TCP could notify Afora whether it is receiving acknowledgments. However, this would allow the possibility that Afora had found a working route to a malicious host masquerading as the desired destination. For this reason, applications that support a notion of authenticity should themselves tell Afora whether they are (or are not) able to positively verify that they are talking to the desired destination. Alternately, a cryptographic protocol layer such as IPSec could inform Afora of whether two-way communication with the correct destination has been established.

Afora could have provided its own cryptographic authentication mechanism, in order to decide when it had found a working route to the authentic owner of the destination host address. However, such an arrangement would be unlikely to provide authentication relevant to many applications. For example, ssh [14] and SFSRO [5] authenticate users and data, respectively, and have limited use for host authentication. Instead, Afora lets applications make the authentication decisions, and concentrates instead on finding routes that satisfy these applications. Further, Afora only requires that source and destination peers authenticate each other; it does not require authentication of the intermediate forwarding nodes.

Afora successfully resists a wide range of malicious or faulty behavior: dropping data or routing protocol packets; forging or modifying routing protocol packets; radio jamming; multiple colluding attackers; attackers on the shortest or fastest path between source and destination; and table exhaustion or bandwidth attacks involving large quantities of routing packets. As long as a safe route exists, Afora will eventually find it despite any combination of these attacks.

Afora has been implemented in the NS [4] simulator. This thesis presents simulation results that explore how long it takes Afora to find safe routes under various circumstances. Afora is relatively slow, since it has to search many paths. This suggests that it should not

be used as a network's main routing protocol, but only when a more conventional routing protocol has failed to find a working route.

1.2 Background: DSR

The Dynamic Source Routing (DSR) protocol [1, 10] is important to this work in two ways. First, the starting point in Afora's design was a much-simplified version of DSR. Second, a DSR-based ad hoc network is a good context in which to consider the potential effects of malicious nodes. We use DSR as an example because it is widely understood, not because it specially or uniquely vulnerable.

A DSR source that wishes to send data to a target must obtain a source route to the target. To do this, the source floods the network with *route request* packets. Each route request packet contains a unique *request identifier*. The original route request packet is received only by the nodes within the radio range of the source. Every node that receives a request forwards it if it is not the target and if it has not yet seen the request's identifier. Each relay node appends its address onto the *route record* of the route request packet before re-broadcasting.

A target that receives a route request sends a *route reply* packet back to the source. This reply contains the route recorded in the request, which the source will use in order to send subsequent data packets to the target. The target returns just the first copy it receives of any particular request. The source receives the reply, extracts the source route, and attaches the source route to subsequent data packets it sends to the target.

A malicious node could successfully attack DSR in a number of ways. If the malicious node is within radio range of the source or target, it can jam them and prevent any communication. If the malicious node is on (or within jamming range of) the route that DSR chooses, the malicious node can prevent data from flowing along that route. A malicious node that isn't near the shortest path could generate incorrect replies in response to the source's queries; if the malicious node is closer than the target is to the source, the malicious node's replies will arrive first and prevent the source from using the correct route provided by the target.

This description has greatly simplified DSR's mechanisms, but it captures the essence required to understand Afora.

1.3 Assumptions and Properties

This section briefly describes the kinds of attacks that Afora can and cannot successfully resist.

The malicious nodes considered in this thesis are assumed to consist of ordinary hardware running software that does not obey the routing protocol. The limitation to ordinary hardware means that the radio transmissions of malicious nodes can only be heard within some finite range. Malicious nodes can take advantage of non-malicious nodes to forward their packets, but only within the rules of the routing protocol, since non-malicious nodes follow those rules. In addition, malicious nodes are assumed not to be able to corrupt non-malicious nodes, except to the extent that the rules of the routing protocol might allow. Malicious nodes are assumed to be able to use jamming to prevent any node within radio range from receiving any packets. These assumptions are meant to capture the capabilities of one or more malicious humans who re-program their laptop or 802.11 radio firmware.

Even with the above restrictions on malicious nodes, there are some attacks that Afora cannot resist. If a malicious node is within radio range of the source or destination, communication between them is not possible (due to jamming). Similarly, if every path between source and target includes a node within radio range of some malicious node, communication is also impossible. It is not likely that any routing protocol could succeed under such conditions.

Afora will eventually succeed in finding a safe path as long as one exists, and as long as malicious nodes satisfy the above assumptions. This means that Afora will resist a very broad spectrum of attacks. Two examples are particularly interesting. First, Afora can find alternate routes around malicious nodes that are on the shortest path between source and destination. Afora probabilistically explores *all* such routes, so it is enough if just one safe route exists, however circuitous. Second, Afora can find those alternate routes even if malicious nodes are generating forged queries or forged replies in response to Afora's

queries, and even if the malicious nodes are closer than the target is to the source.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 describes the Afora algorithm in detail. Chapter 3 argues that Afora will successfully find safe routes despite malicious nodes. Chapter 4 presents simulation results. Chapter 5 outlines future improvements we expect to make to the basic Afora design. Chapter 6 discusses related work. We conclude with Chapter 7.

Chapter 2

Design

Afora is designed to find and use a safe route between a source and a target, if such a route exists. The source node controls Afora's high level operation. The source repeatedly attempts to find a route to the target, until the application indicates that it is able to communicate with the desired target.

When a source wants to find a new route to a target, the source floods a query packet over the whole network. The query packet merely indicates that a route to the target is desired. When the target sees the query packet, it responds by flooding a reply packet over the whole network. The instances of the reply that traverse different routes accumulate source routes. When the source sees a reply, it can use the included source route to send data packets. The key to the protocol's ability to resist malicious attacks is the decision made at each node about whether to forward a reply packet.

A query packet contains a query identifier, the source node's network address, and the target node's network address. A reply packet contains the identifier from the corresponding query packet, the source and target network addresses, and a list of nodes the reply has traversed.

```
struct query <id, source, target>;
struct reply <id, source, target,
              { route }>;
```

The following discussion describes the operation of Afora, organized by the packet

```
source.send_data(target t, data)
  route = routes[t];
  if (route ≠ null and
      app says route is OK)
    send data to t via route;
  else if pending[t] == null
    // look for a new route
    routes[t] = null;
    query q =
      <random(), source, t>;
    pending[t] = q.id;
    broadcast(q);
  end;
end;
```

Figure 2-1: Source send pseudo-code.

processing that takes place in the source, in relay nodes, and in the target. A full justification for the design is deferred until Section 3.

2.1 Initiating the Query

Figure 2-1 shows the pseudo-code that a source uses to send a data packet.

If the source already knows a route to the target (in the `routes` table), and the application has not indicated that it is dissatisfied with the route, the source continues using that route. The application's decision would ideally be made on the basis of cryptographic authentication with a peer application on the target. If the application does not support authentication, it could use a packet loss threshold instead, or arrange for Afora to be driven by TCP timeouts; but in both cases a malicious node could masquerade as the intended target.

If the application is not happy with the current route, the source must find a different route. It formats a query packet and broadcasts it to its neighbors. The source uses the `pending` table to keep track of which targets it is currently querying for.

```
relay.forward_query(query q)
  if qtable[q.source][q.id] == false
    qtable[q.source][q.id] = true;
    broadcast(q);
  end;
end;
```

Figure 2-2: Relay query forwarding pseudo-code.

```
target.receive_query(query q)
  if qtable[q.source][q.id] == false
    qtable[q.source][q.id] = true;
    reply r = <q.id, q.source,
                q.target, {target}>;
    broadcast(r);
  end;
end;
```

Figure 2-3: Target query/reply pseudo-code.

2.2 Forwarding the Query

Figure 2-2 shows the pseudo-code that nodes use to forward query packets. Each node maintains a table (`qtable`) that records which queries it has forwarded; a node only forwards queries it has not already seen. This table is indexed by two keys: the source address and the route query identifier.

2.3 Replying to the Query

Figure 2-3 shows the pseudo-code that a node uses to respond to queries directed to it. A target responds to each query only once, using the same `qtable` mechanism as in Figure 2-2. The target broadcasts a reply packet to its neighbors, initializing the recorded route to contain just the target's address.

```
relay.forward_reply(reply r)
  if rtable[r.source][r.id] == false
  then with probability p
    rtable[r.source][r.id] = true;
    prepend relay to r.route;
    broadcast(r);
  end;
end;
```

Figure 2-4: Relay reply forwarding pseudo-code.

```
source.receive_reply(reply r)
  if (routes[r.target] == null and
      pending[r.target] == r.id)
  then with probability p
    pending[r.target] = null;
    routes[r.target] = r.route;
  end;
end;
```

Figure 2-5: Source reply processing pseudo-code.

2.4 Forwarding the Reply

Figure 2-4 shows hows a node handles a route reply packet.

Each node maintains a table (`rtable`) that records which reply packets it has forwarded. Reply packets with the same source and ID share the same `rtable` entry, so at most one of them will be forwarded, even if they have different recorded routes. If a node receives a reply packet, and it has not already forwarded the packet, then with probability p it broadcasts the packet to its neighbors after adding itself to the recorded route. This means that a node will drop a reply packet with probability $1 - p$ even if it has not seen the reply before. As a result, the query that a node forwards is not always the first reply that arrives.

2.5 Source Reply Processing

Figure 2-5 shows how a node processes replies to query packets that it originated. A source accepts a reply only if the `pending` table indicates that it is waiting for a route with the

target and ID indicated by the reply. The source only accepts the reply with probability p. If a source accepts a reply, it sets its routing table entry for the target to the source-route accumulated in the reply. This means that future data packets sent to the target by the code in Figure 2-1 will use that route.

2.6 Choice of p

The final piece of the design is the choice of p. Afora nodes ignore replies with probability $1 - p$ in order that every possible route between source and destination has a non-zero chance of being the one that the source chooses. This mechanism allows Afora to find safe routes even if malicious nodes control the shortest routes. It also allows Afora to find safe routes even if malicious nodes are generating misleading replies.

p must be less than one; otherwise a malicious node near the source could always arrange for the source to accept its reply first. Smaller values of p have two effects. First, they increase the chance that no legitimate replies at all will be relayed all the way to the source. Second, if malicious nodes close to the source are able to generate many replies, small values of p make it more likely that legitimate nodes will ignore those replies and wait for correct replies. p needs to be low enough that the source has a reasonable probability of ignoring the replies from all but the last neighbor. This means that the choice of p presents a tradeoff between speed in the case that there are no or few malicious nodes, and speed in the case in which there are many malicious nodes sending bogus replies.

Chapter 3

Analysis

The goal of Afora is to ensure that, if a safe route from a source to a target exists, the source will eventually discover it. This chapter argues that Afora achieves this goal, despite malicious nodes' efforts to the contrary.

The source keeps re-querying until it finds a route that delivers packets (see Figure 2-1). As long as each query makes an independent route choice, and each safe route has a non-zero probability of being chosen by each query, Afora will eventually find a safe route. In order to show that each safe route has a non-zero probability of being chosen by each query, the following claims must be true:

1. **Query Liveness:** If there's a safe route, a query packet from the source arrives at the target with non-zero probability.

2. **Reply Liveness:** A reply packet follows each safe route from target back to source with non-zero probability.

3. **Source Choice:** A non-zero fraction of the replies that arrive back at the source contain safe routes.

The chief way in which the claims might be false is if malicious nodes could force the probabilities involved to be zero or vanishingly small, perhaps by flooding source or target with bogus replies. The following sections demonstrate that the claims are true despite such attacks.

3.1 Query Liveness

If there's a safe route, the source's query will follow it to the destination as long as each node on the route forwards a copy of the query. Suppose that node n_1 on a safe route broadcasts a query packet, and that node n_2 is the next node on that route. There are only two ways a malicious node could prevent n_2 from forwarding the query: by disrupting the radio transmission from n_1, or by ensuring that n_2's `qtable` (see Figure 2-2) already has an entry for the query.

Since the route in question is safe, none of the nodes on the route are malicious, and no node within radio range of the route is malicious. This means that a malicious node cannot directly jam n_1's transmission.

A malicious node might try to generate large volumes of traffic that non-malicious nodes will forward into radio range of n_2. The wireless MAC's fairness mechanisms will likely prevent this from being an effective denial-of-service attack; if they do not, the mechanism proposed in Section 5.1 will work.

Finally, n_2 will not forward the query if its `qtable` already has an entry for this query. The only way for a malicious node to cause n_2 to have a `qtable` entry for the query is for the malicious node to have already sent an identical query. The query must be identical, since the `qtable` is indexed by all fields of the query. In this case, node n_2 will have already forwarded a copy of the query. The only reason this could be undesirable is if the malicious node forged the query before the source sent it out. Random choice of query identifiers from a large identifier space makes it very unlikely that an attacker could guess the source's query ID in advance.

3.2 Reply Liveness

The previous section showed that a malicious node cannot reliably prevent a query from reaching the target, as long as a safe route exists. The target code (Figure 2-3) always produces a broadcast reply packet in response to a query. To see that a copy of the reply will be forwarded along each safe route back to the source with non-zero probability, consider

one of the safe routes. It is enough if each node along the route forwards a copy of the reply with non-zero probability. Again, suppose that n_1 broadcasts a copy of the reply, and that n_2 is the next node on the safe route (towards the source).

For the reasons mentioned in the previous section, a malicious node cannot directly or indirectly interfere with the reply transmission, and thus cannot prevent n_2 from receiving a copy of the reply packet.

n_2 will forward the reply with non-zero probability p unless n_2 already has an entry for the reply in its `rtable` (see Figure 2-4). A malicious node could cause this to be true by sending bogus replies (with correct source and id fields) to neighbors of n_2. Since those neighbors are non-malicious, each will forward at most one bogus reply, due to the neighbor's `rtable`. Thus the number of bogus replies that n_2 receives before it receives the safe reply from n_1 is limited by the number of neighbors it has. Since n_2 drops each of those bogus replies with probability $1 - p$, there is a non-zero probability that n_2 will drop all the bogus replies and forward the safe reply. The key observation is that a malicious node has limited ability to influence this probability.

3.3 Source Choice

The previous section showed that a reply will arrive at the source from each safe route with non-zero probability. The source accepts only one reply, chosen randomly. In order that it try a safe route with non-zero probability, it must be the case that the number of bad routes it receives is bounded. The argument for this property is the same as in the previous section: since the source's neighbors are non-malicious, the total number of replies a source will receive is bounded by its number of neighbors. Thus malicious nodes cannot force the probability of the source choosing a safe route arbitrarily close to zero.

Chapter 4

Experiments

While Chapter 3 demonstrated that Afora will eventually find a safe route if one exists, it did not describe how long that process would take. This Chapter uses *ns* version 2.1b8a [4] to explore the number of queries it takes Afora to find a safe route in various circumstances. The simulator implements the design as described in Chapter 2.

The simulator implements Afora over the 802.11 MAC layer, with a fixed radio range of 250 meters.

Unless noted, the simulation parameters are as follows. The simulated universe is 800 meters by 800 meters. The total number of nodes is 50, which turns out to put each node in radio range of an average of 15 other nodes. The nodes do not move. The default value of p is 0.5.

Each data point in the graphs below summarizes 400 simulation runs with different random number generator seeds. Most of the graphs present the median, 10th, and 90th percentiles of the runs in each configuration.

In each simulation run, there is one source and one target in the network. The source is randomly placed in the left-hand quarter of the square universe. The target is randomly placed in the right-hand quarter of the universe. The rest of the 48 relay nodes are randomly placed across the universe.

In each simulation run, the source keeps re-querying until it finds a safe route to the destination; at that point the simulation run is over. The source gives up after 720 tries. Most of the graphs shown below include only the runs that produced a safe route; the

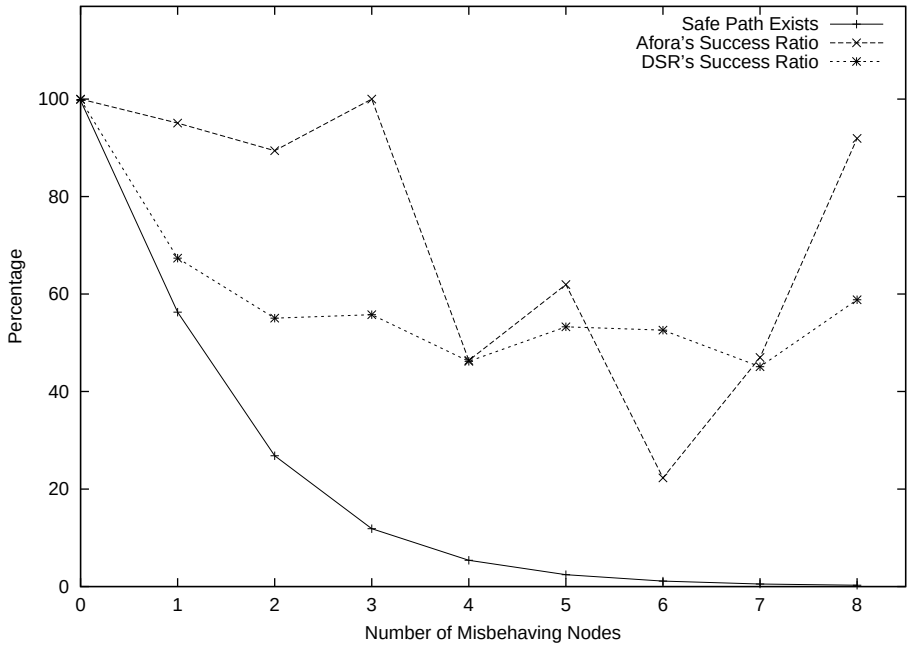

Figure 4-1: The percentage of runs in which a safe path exists, as a function of the number of passive malicious nodes. The graph also includes, for the runs that have a safe path, the percentage of such runs in which Afora and DSR, respectively, find a safe path. The data are from the same simulations as Figure 4-2.

unsuccessful runs are discussed separately.

We simulate two kinds of malicious nodes (each experimental description will describe which was used). A *passive* malicious node forwards query and reply packets correctly, but drops all data packets and jams its neighbors' radios. An *active* malicious node generates incorrect reply packets whenever it receives a query packet, as well as dropping and jamming data. Such a node generates 50 replies for each query it receives; each reply contains a different (but incorrect) path to the desired target. Malicious nodes are randomly chosen from the pool of 48 nodes.

4.1 Number of Passive Malicious Nodes

Figures 4-1 and 4-2 show the effect of varying the number of passive malicious nodes.

Figure 4-1 shows the fraction of simulation runs in which a safe route existed, and of those runs, the fraction in which Afora and DSR found a safe route, respectively. As

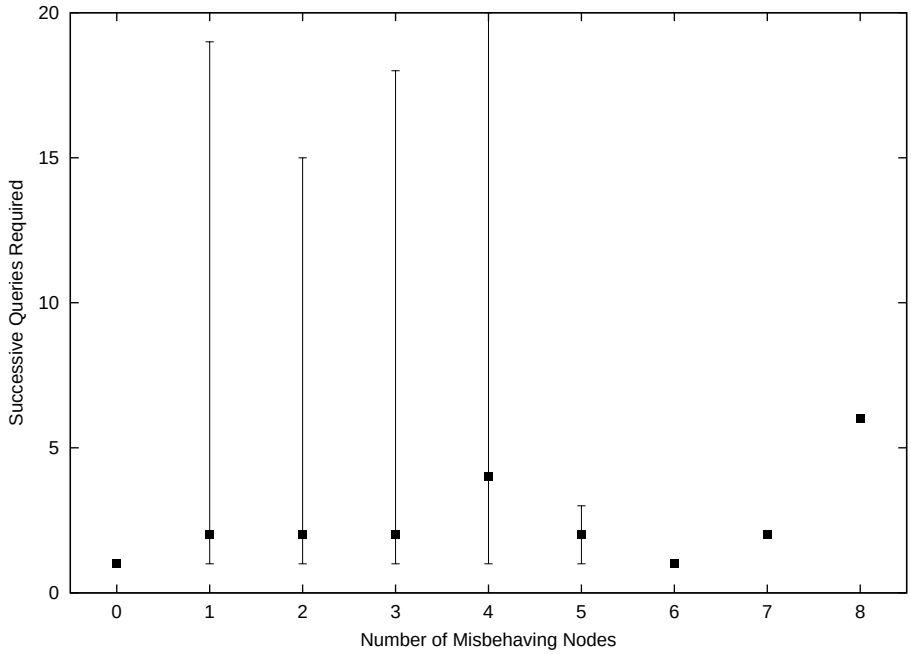

Figure 4-2: The median number of queries required to find a safe path, as a function of the number of passive malicious nodes. The error bars indicate 10th and 90th percentiles; the 90th percentile for 4 nodes is 80. Only successful runs are included here; Figure 4-1 shows what fraction of runs are successful.

the number of malicious nodes increases, the total expected area covered by their radios increases, and the likelihood of even a single safe route existing decreases. Figure 4-1 shows that with three or fewer malicious nodes, Afora finds a safe route in nearly all cases in which one exists. As a base-line for comparison, DSR (which was not designed to resist malicious behavior) finds safe routes much less often than Afora; DSR effectively finds safe routes in just the cases in which the shortest path is safe.

Figure 4-2 shows the number of queries it takes Afora to find a safe route provided one exists. The y value represents the median number of queries required to find a safe route. The error bars represent 10th and 90th percentiles.

When there are no malicious nodes, Afora almost always finds a route with just one query. As the number of malicious nodes increases, the number of safe routes decreases, and more queries are typically required to find one. With five or more malicious nodes, however, safe routes become so rare that results concerning those few that are found are not very meaningful. With 5 or more malicious nodes, the occasional safe route that does exist tends to be a short, direct and easy to find route from source to target, since longer routes are extremely likely to encounter a malicious node on their path. This explains why it is comparatively easier and faster to find shorter routes that do exist as the number of malicious nodes grows beyond 5.

4.2 Total Number of Nodes

Figure 4-3 shows the effect of varying the total number of nodes. The size of the universe is varied to keep the average node density constant. 2% of the nodes are passively malicious.

As the number of nodes grows, Afora requires more queries to find a safe route. This is because the routes have more hops in large systems, and a reply's probability of being dropped by the p mechanism increases with the number of hops. On the other hand, the number of ways a reply can travel from destination to source increases with the total number of nodes. These two effects partially offset each other.

Approximately 50% of the simulation results in this experiment were omitted from Figure 4-3 as no safe routes were found in those simulations.

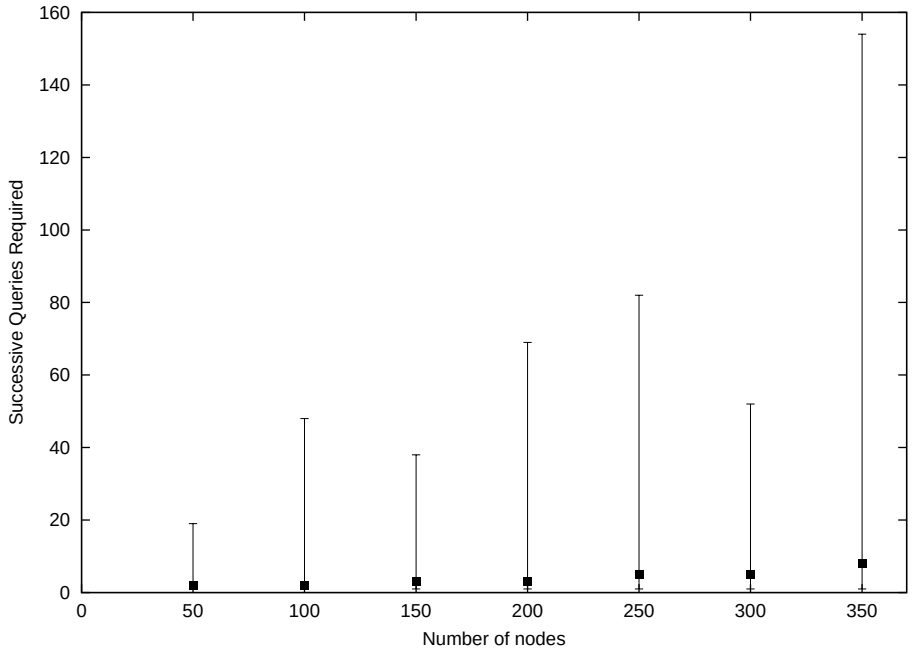

Figure 4-3: The median number of successive queries required to find a safe path, as a function of the total number of nodes. The error bars indicate 10th and 90th percentiles. 2% of the nodes are passively malicious.

Figure 4-3 shows that even as the network grows up to 350 nodes, the median number of queries required by an Afora source stays low.

4.3 Effect of Active Misbehavior

Figures 4-4 and 4-5 show the effects of varying the number of *active* malicious nodes.

The results are similar to those obtained in Section 4.1, except that the active nodes make it harder to find a safe route. Each malicious node generates enough fake replies that its neighbors are very likely to forward a few of them. A query only succeeds if all the bad replies are dropped by the p mechanism and at least one good reply isn't dropped; the probability of this is not high. As a result, Afora needs to send more queries to find a safe route, as Figure 4-5 shows.

As in Section 4.1, only a very small fraction of simulation runs are successful with four or more malicious nodes. For the same reasons, the right half of Figure 4-5 is not very

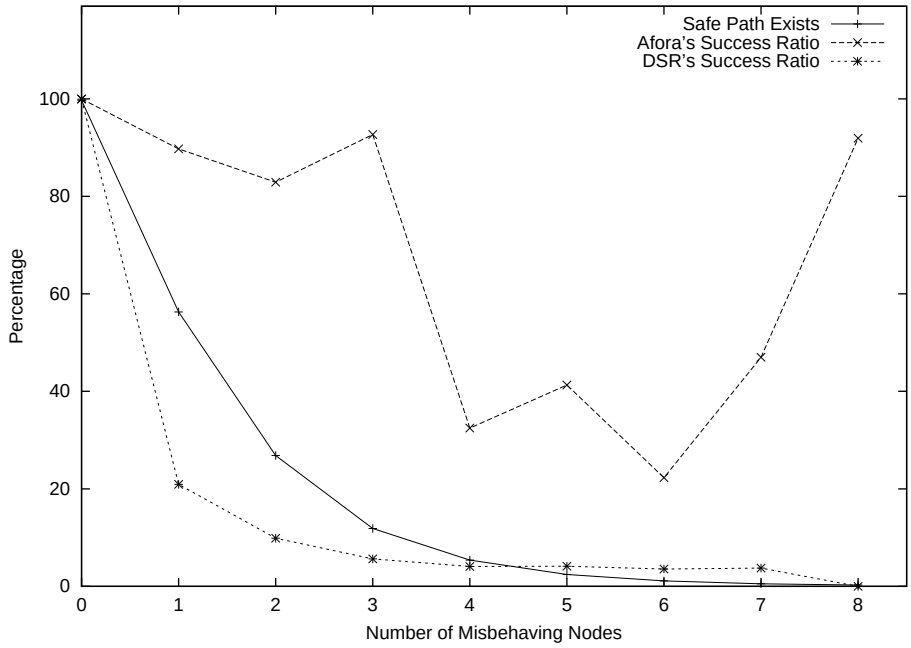

Figure 4-4: The percentage of runs in which a safe path exists, as a function of the number of active malicious nodes. The graph also includes, for the runs that have a safe path, the percentage of such runs in which Afora and DSR, respectively, find a safe path. The data are from the same simulations as Figure 4-5.

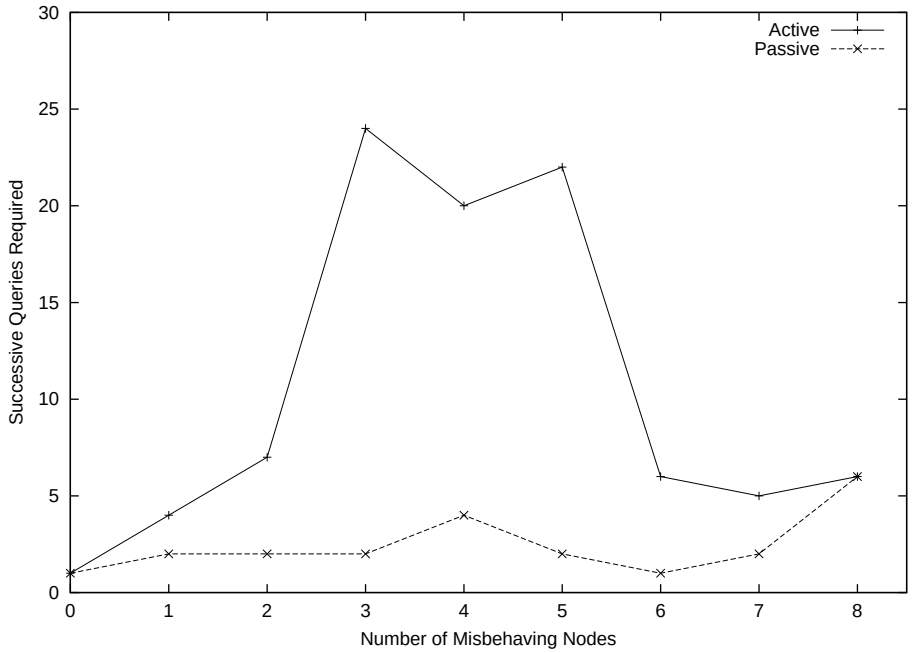

Figure 4-5: The median successive queries required to find a safe path, as a function of number of malicious nodes. Data for both passive and active malicious nodes are shown. There are 50 nodes in each experiment and p is 0.5.

meaningful, and the occasional routes that do exist tend to be short and easy to find.

Though the numbers of queries in Figure 4-5 may seem large for active adversaries, recall that we are placing few restrictions on the malicious nodes. We are allowing a significant fraction of the nodes to be malicious. The scenarios make it likely that there are multiple malicious nodes closer to the source than the destination is, and thus able to get fake replies to the source before the real replies can arrive. The malicious nodes are allowed to generate a large volume of fake replies; in real life this would be pretty obvious and the malicious nodes could easily be detected.

4.4 Choice of p

Figure 4-6 shows the effect of varying p on the median number of queries it takes for the source to find a safe path. One of the 50 nodes in each simulation is actively malicious.

When p is near 1.0, the most likely outcome is that the source receives a reply after a

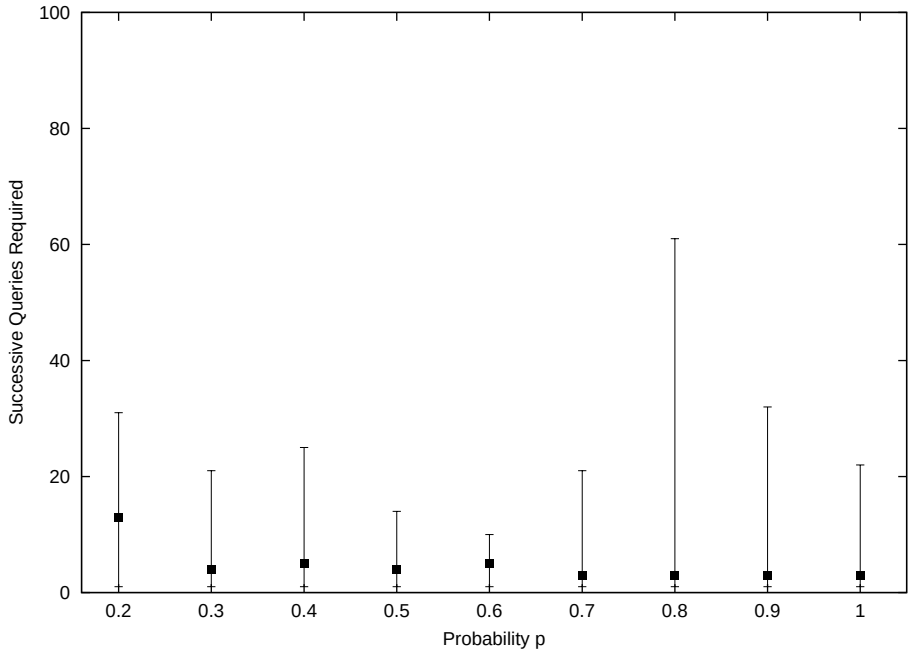

Figure 4-6: The median number of successive queries required to find a safe path, as a function of p. The error bars indicate 10th and 90th percentiles. There are 50 nodes in total in each simulation, of which one is actively malicious.

single query, but that the reply is likely to have originated from the malicious node. The source must try many queries before it finds a path that is safe. This explains the higher 90th percentile when p approaches 1.0. When p is very small, the probability is high that every reply (good and bad) is dropped before reaching the source. At intermediate values of p there is a noticeable chance that all the bad replies are dropped, leaving an opportunity for a good reply to arrive.

Chapter 5

Future Work

This chapter outlines a few ways in which we expect to improve the basic Afora design.

5.1 Avoiding Table Exhaustion

The `qtable` and `rtable` are potentially vulnerable to denial-of-service attacks in which a malicious node generates large numbers of query or reply packets, each with a different source and/or query ID. In the design as presented in Chapter 2, some fraction of these would be flooded over the whole network, consuming table space and bandwidth. A sufficiently energetic attacker could exhaust the network's resources.

The basic Afora design does provide weak resistance to this kind of attack, since it probabilistically drops some replies. The farther a node is from an attacker, the fewer bogus packets the node will receive. We can strengthen this idea into a complete defense: each node should simply limit the rate at which it is willing to forward query or reply packets to some relatively small number of packets per second.

As a result, if a malicious node generates randomized query or reply packets, the nodes next to it will be rendered useless; their allowed forwarding rate will be completely consumed by the bogus packets. (This is not a new problem, since the malicious node could directly jam their radios.) However, the neighbor nodes will only forward the bogus packets at the allowed rate. Thus nodes more than one hop away from the malicious nodes receive the bogus packets at a bounded rate. Since the allowed rate is known, the rate, the

maximum table sizes, and the table entry lifetimes can be chosen together to ensure that the tables never overflow. In addition, the bounded rate of bogus packets means that the p mechanism is likely to choose more or less fairly between bogus packets and valid packets.

5.2 Reduced Communication Costs

Since the flooding process used to obtain a route can potentially involve far more nodes than would be part of the route itself, and it could have to be repeated several times before a safe route is obtained, there exists the risk of the route selection protocol becoming the main consumer of bandwidth in the network. It is therefore important to attempt to reduce the amount of communication involved to a minimum.

Loop formation is automatically avoided by Afora since every relay node propagates only one route; because we do not need full routes at relay nodes to check for loops, we can then avoid sending the full route (the largest amount of information broadcast in the protocol for large networks) back in `target.receive_query` and `relay.forward_reply`. Upon hearing a node x broadcasting a reply, a relay node y need only store the identity of x in a table `next`[] indexed like the `qtable` and the `rtable`, and then, in place of the full route from the target to itself, broadcast only its own identity. Only if and when the source decides to use a route from a particular neighbor, it can ask that neighbor for the rest of the route - that neighbor can in turn recursively ask the rest of the route to the next node, until the target (identified by a null `next`[] entry) is reached. Or the source could directly use the route "implicitly" without learning it, having every node sending data to the next node together with the route id, in a way similar to that proposed by [7].

5.3 Adapting p

As described in Section 2.6, the best value for p depends on whether there happen to be malicious nodes on the shortest path or near the source, on whether malicious nodes are generating bogus replies, and on the number of neighbors each node has. Thus it makes sense for the value of p to be chosen dynamically for each query by the source.

The source can spread the desired value of p by making it part of the query id field. In this way, it can guarantee that malicious nodes will not be able to tamper with it. In fact, queries and replies with different IDs will not interfere with each other on the target and relay nodes, and the source will accept only replies with the same id (and therefore p) as the last query to that target it has issued itself. Then any malicious node who changes the value of p in a query/reply will be effectively dropping that query/reply and broadcasting a different one, an attack we have already shown (in 5.1) to be ineffective.

A heuristic for adaptively choosing an appropriate value of p which seems to be effective is the following:

- Begin with $p = 1$. This guarantees best performance in the absence of misbehaving nodes.
- Whenever the source sends a query and does not obtain any route, increase p by 0.05. Whenever the source sends a query and receives a route which proves to be ineffective, decrease p by 0.05.

In this way the source, if it does not find a safe route quickly, approaches a steady state where it will obtain a route on about half the queries, thus striking a balance between allowing a fair probability that short but unsafe routes be discarded and actually obtaining a sufficient number of new routes to try.

Chapter 6

Related Work

Zhou and Haas [18] identified five attributes of a secure ad hoc network: availability, confidentiality, integrity, authentication, and non-repudiation. To satisfy the first attribute (availability), they acknowledged that routing protocols must be secure, and suggested that taking advantage of multiple routes between nodes is important in defending against security threats. Afora is a realization of this idea.

Marti et al. propose a technique that identifies misbehaving nodes in an ad hoc network and selects routes that avoid those nodes [12]. A node detects misbehavior in one of its neighbors by observing that the neighbor fails to forward data packets. This scheme works well as long as malicious nodes are limited to dropping data packets; it may not work if nodes generate malicious protocol packets. For example, a malicious node could generate DSR replies with source routes that include the malicious node and then a non-existent node. The last non-malicious node would observe the malicious node forwarding data packets to the supposed next node, and believe that all was well. Afora resists a wider class of attacks, though it is less efficient.

The Ariadne [8] routing protocol, which is based on DSR, prevents malicious nodes from tampering with routing queries and replies. It uses hop-by-hop symmetric cryptography to protect the routing packets; as a result it is far more efficient at this task than Afora. However, Ariadne makes some compromises to achieve its efficiency. Ariadne does not systematically explore the network for alternate uncompromised paths, so it is vulnerable to a malicious node on or near the shortest path. Ariadne requires that every node share

cryptographic key material with every other node; this is a source of complexity as well as potential compromise. Ariadne assumes the existence of a secure time synchronization facility. Afora, while slower, is more general in the sense that it does explore for alternate paths and makes fewer assumptions.

The Ariadne authors introduce an attacker taxonomy useful in judging a protocol's resistance to malicious nodes. In their terms Afora is secure against ActiveX, ActiveC, ActiveCX, and ActiveCCX attacks. These are attacks in which attackers inject malicious packets into the network, and in which attackers know the per-node cryptographic keys of many nodes (but not the source or destination nodes). Afora resists the latter attacks in the sense that it does not depend on intermediate nodes having cryptographic keys. The only type of attack Afora cannot defend against is ActiveVC, in which attackers control all paths from source to target.

The SAR protocol views secure ad hoc routing as a Quality-of-Service issue [16]. SAR groups nodes into various levels of trust, and allow only nodes with the same trust level to form a route. SAR nodes at the same trust level share an encryption key. SAR, unlike Afora, requires nodes at each trust level to trust each other.

ARAN [3] is an on-demand, AODV-like protocol that provides authentication, integrity, and non-repudiation. ARAN relies on the existence of a trusted service to issue certificates to all participating network nodes. ARAN nodes use nested hop-by-hop signatures to protect query and reply packets. While ARAN prevents a malicious node that is not on the shortest path from re-routing packets through itself, it does not help in cases where the malicious node is on the shortest path. Afora, in contrast, can handle malicious nodes on the shortest path.

Hubaux et al. suggest the use of a virtual currency to promote cooperation among nodes [9, 2]. Nodes in the ad hoc network pay each other with the currency for forwarding packets. Tamper-resistant hardware in each node maintains the integrity of the currency, and prevents forgery and re-use.

Guerrero proposes SAODV, an extension to AODV [17], which authenticates AODV control packets using digital signatures, and verifies the hop counts using a hash chain [11]. SAODV assumes that every node in the ad hoc network has some way of obtaining the

public keys of all other nodes.

An on-demand protocol proposed by Papadimitratos and Haas, SRP [13], assumes the existence of a shared secret between each source and destination. Source and destination authenticate query and reply messages using MACs keyed with the shared secret. Multiple queries traversing over distinct routes can reach the destination. While SRP prevents malicious nodes from modifying or forging routing packets, it does not attempt to avoid nodes that fail to forward data packets properly.

The idea of having all nodes in an ad hoc network probabilistically drop a fraction of the packets they receive has recently received attention in the ad hoc networking community, but for purposes different from ours. For example, Vadhat and Becker [15] study it in connection with unicast routing attempting to ensure that messages are eventually delivered even if there is never a connected path between source and target at any given point in time. Haas, Alpern and Li [6] propose it as a technique to reduce total traffic when flooding a message over a dense network. To the best of our knowledge, this is the first time this technique is used to ensure communication in the face of malicious nodes.

Chapter 7

Conclusion

The Afora protocol finds routes in ad hoc networks despite the presence of misbehaving nodes. It can defend against attackers that drop packets, jam radio reception, and forge or modify routing protocol packets. If path exists that is outside of jamming range of any attacker, Afora will eventually find it.

Afora has a number of limitations. It cannot defend against attackers with radio ranges that cover all routes between a source and destination, or against attackers who can break into innocent nodes and corrupt them. It requires some higher network layer to tell it whether data is being delivered to the correct recipient; this essentially means the higher layer must use end-to-end cryptographic authentication. A final limitation is that Afora provides probabilistic guarantees; it proceeds in a number of rounds, each of which has some probability of resulting in a safe route. In large networks the number of rounds is likely to be large.

Bibliography

[1] Josh Broch, David Johnson, and David Maltz. The Dynamic Source Routing protocol for mobile ad hoc networks. Internet draft (work in progress), Internet Engineering Task Force, October 1999. `http://www.ietf.org/internet-drafts/draft-ietf-manet-dsr-03.txt`.

[2] L. Buttyán and J.-P. Hubaux. Enforcing Service Availability in Mobile Ad Hoc WANs. In *Proc. ACM/IEEE MobiHoc*, August 2000.

[3] B. Dahill, B. Levine, E. Royer, and C. Shields. A Secure Routing Protocol for Ad Hoc Networks. Technical report um-cs-2001-037, University of Massachusetts, Amherst, MA, August 2001.

[4] Kevin Fall and Kannan Varadhan. *ns* notes and documentation. Technical report, UC Berkeley, LBL, USC/ISI, and Xerox PARC, November 1997.

[5] Kevin Fu, M. Frans Kaashoek, and David Mazières. Fast and secure distributed read-only file system. In *Proceedings of the 4th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*, pages 181–196, October 2000.

[6] Z. J. Haas, J. Y. Halpern, and L. Li. Gossip-based ad hoc routing. In *Proc. IEEE Infocom*, July 2002.

[7] Yih-Chun Hu and David Johnson. Implicit source routes for on-demand ad hoc network routing. In *Proc. ACM/IEEE MobiHoc*, October 2001.

[8] Yih-Chun Hu, Adrian Perrig, and David B. Johnson. Ariadne: A secure on-demand routing protocol for ad hoc networks. Technical Report TR01-383, Rice University, March 2002.

[9] J.-P. Hubaux, L. Buttyán, and S. Čapkun. The Quest for Security in Mobile Ad Hoc Networks. In *Proc. ACM/IEEE MobiHoc*, October 2001.

[10] David B. Johnson. Routing in ad hoc networks of mobile hosts. In *Proc. of the IEEE Workshop on Mobile Computing Systems and Applications*, pages 158–163, December 1994.

[11] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, November 1981.

[12] S. Marti, T. Giuli, K. Lai, and M. Baker. Mitigating Routing Misbehavior in Mobile Ad Hoc Networks. In *Proc. ACM/IEEE MobiCom*, pages 255–265, August 2000.

[13] Panagiotis Papadimitratos and Zygmunt J. Haas. Secure routing for mobile ad hoc networks. In *Proc. SCS Communication Networks and Distibuted Systems Modeling and Simulation Conference (CNDS'02)*, January 2002.

[14] T. Rinne, T. Ylonen, T. Kivinen, M. Saarinen, and S. Lehtinen. SSH authentication protocol. `http://www.ietf.org/internet-drafts/draft-ietf-secsh-userauth-13.txt`, November 2001.

[15] A. Vahdat and D. Becker. Epidemic routing for partially connected ad hoc networks, 2000.

[16] Seung Yi, Prasad Naldurg, and Robin Kravets. Security-aware ad hoc routing for wireless networks. Technical report, University of Illinois, Urbana-Champaign, August 2001. UIUCDCS-R-2001-2241.

[17] Manel Guerrero Zapata. Secure ad hoc on-demand distance vector (SAODV) routing. Internet draft sent to manet@itd.nrl.navy.mil mailing list, August 2001.

[18] L. Zhou and Z. J. Haas. Securing ad hoc networks. *IEEE Network Magazine*, 13(6), November 1999.