Mitigating Routing Misbehavior in Mobile Ad Hoc Networks

Sergio Marti, T.J. Giuli, Kevin Lai, and Mary Baker
Department of Computer Science
Stanford University
Stanford, CA 94305 U.S.A
{smarti,giuli,laik,mgbaker}@stanford.edu

ABSTRACT

This paper describes two techniques that improve throughput in an ad hoc network in the presence of nodes that agree to forward packets but fail to do so. To mitigate this problem, we propose categorizing nodes based upon their dynamically measured behavior. We use a *watchdog* that identifies misbehaving nodes and a *pathrater* that helps routing protocols avoid these nodes. Through simulation we evaluate watchdog and pathrater using packet throughput, percentage of overhead (routing) transmissions, and the accuracy of misbehaving node detection. When used together in a network with moderate mobility, the two techniques increase throughput by 17% in the presence of 40% misbehaving nodes, while increasing the percentage of overhead transmissions from the standard routing protocol's 9% to 17%. During extreme mobility, watchdog and pathrater can increase network throughput by 27%, while increasing the overhead transmissions from the standard routing protocol's 12% to 24%.

1. INTRODUCTION

There will be tremendous growth over the next decade in the use of wireless communication, from satellite transmission into many homes to wireless personal area networks. As the cost of wireless access drops, wireless communications could replace wired in many settings. One advantage of wireless is the ability to transmit data among users in a common area while remaining mobile. However, the distance between participants is limited by the range of transmitters or their proximity to wireless access points. Ad hoc wireless networks mitigate this problem by allowing out of range nodes to route data through intermediate nodes.

Ad hoc networks have a wide array of military and commercial applications. Ad hoc networks are ideal in situations where installing an infrastructure is not possible because the infrastructure is too expensive or too vulnerable, the network is too transient, or the infrastructure was destroyed. For example, nodes may be spread over too large an area for one base station and a second base station may be too expensive. An example of a vulnerable infrastructure is a military base station on a battlefield. Networks for wilderness expeditions and conferences may be transient if they exist for only a short period of time before dispersing or moving. Finally, if network infrastructure has been destroyed due to a disaster, an ad hoc wireless network could be used to coordinate relief efforts. Since DARPA's PRNET [13], the area of routing in ad hoc networks has been an open research topic.

Ad hoc networks maximize total network throughput by using all available nodes for routing and forwarding. Therefore, the more nodes that participate in packet routing, the greater the aggregate bandwidth, the shorter the possible routing paths, and the smaller the possibility of a network partition. However, a node may misbehave by agreeing to forward packets and then failing to do so, because it is overloaded, selfish, malicious, or broken. An overloaded node lacks the CPU cycles, buffer space or available network bandwidth to forward packets. A selfish node is unwilling to spend battery life, CPU cycles, or available network bandwidth to forward packets not of direct interest to it, even though it expects others to forward packets on its behalf. A malicious node launches a denial of service attack by dropping packets. A broken node might have a software fault that prevents it from forwarding packets.

Misbehaving nodes can be a significant problem. Our simulations show that if 10%-40% of the nodes in the network misbehave, then the average throughput degrades by 16%-32%. However, the worst case throughput experienced by any one node may be worse than the average, because nodes that try to route through a misbehaving node experience high loss while other nodes experience no loss. Thus, even a few misbehaving nodes can have a severe impact.

One solution to misbehaving nodes is to forward packets only through nodes that share an *a priori* trust relationship. *A priori* trust relationships are based on pre-existing relationships built outside of the context of the network (e.g. friendships, companies, and armies). The problems with relying on *a priori* trust-based forwarding are that 1) it requires key distribution, 2) trusted nodes may still be overloaded, 3) trusted nodes may still be broken, 4) trusted nodes may be compromised, and 5) untrusted nodes may be well behaved. It may not be possible to exchange keys used to authenticate trusted nodes outside of the ad hoc network before the conference or disaster that requires an ad hoc network. If keys are not distributed ahead of time, then enforcing *a priori* trust-based forwarding requires a secure channel for key exchanges within the ad hoc network for authentication. Even if keys can be exchanged, a trusted node's security may be compromised, or a trusted node may be overloaded or broken as mentioned above. Finally, although relying on *a priori* trust-based forwarding reduces the number of misbehaving nodes, it also excludes untrusted well behaved nodes whose presence could improve ad hoc network performance.

Another solution to misbehaving nodes is to attempt to forstall or isolate these nodes from within the actual routing protocol for the network. However, this would add significant complexity to protocols whose behavior must be very well defined. In fact, current versions of mature ad hoc routing algorithms, including DSR [12], AODV [7], TORA [5], DSDV [19], STAR [9], and others [16] only detect if the receiver's network interface is accepting packets, but they otherwise assume that routing nodes do not misbehave. Although trusting all nodes to be well behaved increases the number of nodes available for routing, it also admits misbehaving nodes to the network.

In this paper we explore a different approach, and install extra facilities in the network to to detect and mitigate routing misbehavior. In this way, we can make only minimal changes to the underlying routing algorithm. We introduce two extensions to the Dynamic Source Routing algorithm (DSR) [12] to mitigate the effects of routing misbehavior: the *watchdog* and the *pathrater*. The watchdog identifies misbehaving nodes, while the pathrater avoids routing packets through these nodes. When a node forwards a packet, the node's watchdog verifies that the next node in the path also forwards the packet. The watchdog does this by listening *promiscuously* to the next node's transmissions. If the next node does not forward the packet, then it is misbehaving. The pathrater uses this knowledge of misbehaving nodes to choose the network path that is most likely to deliver packets.

Using the ns network simulator [8], we show that the two techniques increase throughput by 17% in the presence of up to 40% misbehaving nodes during moderate mobility, while increasing the ratio of overhead transmissions to data transmissions from the standard routing protocol's 9% to 17%. During extreme mobility, watchdog and pathrater can increase network throughput by 27%, while increasing the percentage of overhead transmissions from 12% to 24%. We describe mechanisms to reduce this overhead in Section 6.

The remainder of this paper is organized as follows. Section 2 specifies our assumptions about ad hoc networks and gives background information about DSR. Section 3 describes the watchdog and pathrater extensions. Section 4 describes the methodology we use in our simulations and the metrics we use to evaluate the results. We present these results in Section 5. Sections 6 and 7 present related work and future work, respectively. Finally, Section 8 concludes the paper.

2. ASSUMPTIONS AND BACKGROUND

This section outlines the assumptions we make regarding the properties of the physical and network layers of ad hoc networks and includes a brief description of DSR, the routing protocol we use.

2.1 Definitions

We use the term *neighbor* to refer to a node that is within wireless transmission range of another node. Likewise, *neighborhood* refers to all the nodes that are within wireless transmission range of a node.

2.2 Physical Layer Characteristics

Throughout this paper we assume bidirectional communication symmetry on every link between nodes. This means that if a node B is capable of receiving a message from a node A at time t, then node A could instead have received a message from node B at time t. This assumption is often valid, since many wireless MAC layer protocols, including IEEE 802.11 and MACAW [2], require bidirectional communication for reliable transmission. The watchdog mechanism relies on bidirectional links.

In addition, we assume wireless interfaces that support *promiscuous* mode operation. Promiscuous mode means that if a node A is within range of a node B, it can overhear communications to and from B even if those communications do not directly involve A. Lucent Technologies' WaveLAN interfaces have this capability. While promiscuous mode is not appropriate for all ad hoc network scenarios (particularly some military scenarios) it is useful in other scenarios for improving routing protocol performance [12].

2.3 Dynamic Source Routing (DSR)

DSR is an on-demand, source routing protocol. Every packet has a route path consisting of the addresses of nodes that have agreed to participate in routing the packet. The protocol is referred to as "on-demand" because route paths are discovered at the time a source sends a packet to a destination for which the source has no path.

We divide DSR into two main functions: route discovery

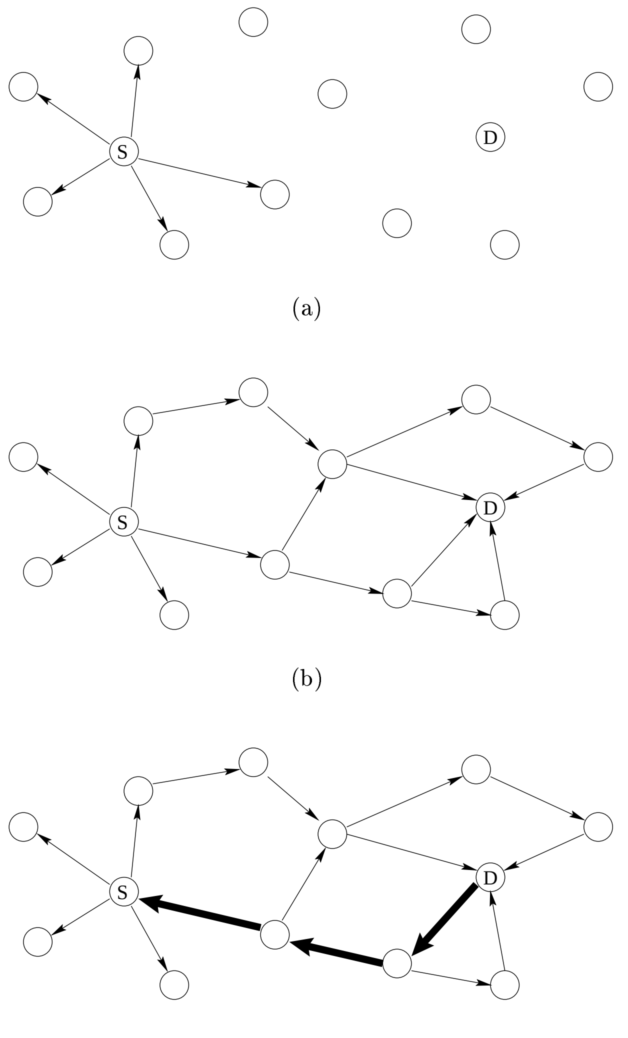


Figure 1: Example of a ROUTE REQUEST. (a) Node S sends out a ROUTE REQUEST packet to find a path to node D. (b) The ROUTE REQUEST is forwarded throughout the network, each node adding its address to the packet. (c) D then sends back a ROUTE REPLY to S using the path contained in one of the ROUTE REQUEST packet that reached it. The thick lines represent the path the ROUTE REPLY takes back to the sender.

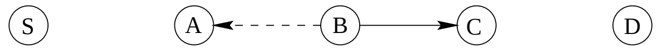


Figure 2: When B forwards a packet from S toward D through C, A can overhear B's transmission and can verify that B has attempted to pass the packet to C. The solid line represents the intended direction of the packet sent by B to C, while the dashed line indicates that A is within transmission range of B and can overhear the packet transfer.

and route maintenance. Figure 1 illustrates route discovery. Node S (the source) wishes to communicate with node D (the destination) but does not know any paths to D. S initiates a route discovery by broadcasting a ROUTE REQUEST packet to its neighbors that contains the destination address D. The neighbors in turn append their own addresses to the ROUTE REQUEST packet and rebroadcast it. This process continues until a ROUTE REQUEST packet reaches D. D must now send back a route reply packet to inform S of the discovered route. Since the ROUTE REQUEST packet that reaches D contains a path from S to D, D may choose to use the reverse path to send back the reply (bidirectional links are required here) or to initiate a new route discovery back to S. Since there can be many routes from a source to a destination, a source may receive multiple route replies from a destination. DSR caches these routes in a route cache for future use.

The second main function in DSR is route maintenance, which handles link breaks. A link break occurs when two nodes on a path are no longer in transmission range. If an intermediate node detects a link break when forwarding a packet to the next node in the route path, it sends back a message to the source notifying it of that link break. The source must try another path or do a route discovery if it does not have another path.

3. WATCHDOG AND PATHRATER

In this section we present the watchdog and the pathrater — tools for detecting and mitigating routing misbehavior. We also describe the limitations of these methods. Though we implement these tools on top of DSR, some of our concepts can be generalized to other source routing protocols. We note those concepts that can be generalized during our descriptions of the techniques.

3.1 Watchdog

The watchdog method detects misbehaving nodes. Figure 2 illustrates how the watchdog works. Suppose there exists a path from node S to D through intermediate nodes A, B, and C. Node A cannot transmit all the way to node C, but it can listen in on node B's traffic. Thus, when A transmits a packet for B to forward to C, A can often tell if B transmits the packet. If encryption is not performed separately for each link, which can be expensive, then A can also tell if B has tampered with the payload or the header.

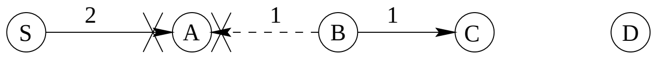


Figure 3: Node A does not hear B forward packet 1 to C, because B's transmission collides at A with packet 2 from the source S.

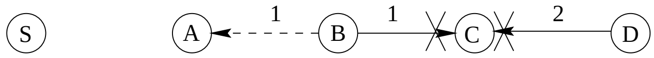


Figure 4: Node A believes that B has forwarded packet 1 on to C, though C never received the packet due to a collision with packet 2.

We implement the watchdog by maintaining a buffer of recently sent packets and comparing each overheard packet with the packet in the buffer to see if there is a match. If so, the packet in the buffer is removed and forgotten by the watchdog, since it has been forwarded on. If a packet has remained in the buffer for longer than a certain timeout, the watchdog increments a failure tally for the node responsible for forwarding on the packet. If the tally exceeds a certain threshold bandwidth, it determines that the node is misbehaving and sends a message to the source notifying it of the misbehaving node.

The watchdog technique has advantages and weaknesses. DSR with the watchdog has the advantage that it can detect misbehavior at the forwarding level and not just the link level. Watchdog's weaknesses are that it might not detect a misbehaving node in the presence of 1) ambiguous collisions, 2) receiver collisions, 3) limited transmission power, 4) false misbehavior, 5) collusion, and 6) partial dropping.

The ambiguous collision problem prevents A from overhearing transmissions from B. As Figure 3 illustrates, a packet collision can occur at A while it is listening for B to forward on a packet. A does not know if the collision was caused by B forwarding on a packet as it should or if B never forwarded the packet and the collision was caused by other nodes in A's neighborhood. Because of this uncertainty, A should not immediately accuse B of misbehaving, but should instead continue to watch B over a period of time. If A repeatedly fails to detect B forwarding on packets, then A can assume that B is misbehaving.

In the receiver collision problem, node A can only tell whether B sends the packet to C, but it cannot tell if C receives it (Figure 4). If a collision occurs at C when B first forwards the packet, A only sees B forwarding the packet and assumes that C successfully receives it. Thus, B could skip retransmitting the packet and leave A none the wiser. B could also purposefully cause the transmitted packet to collide at C by waiting until C is transmitting and then forwarding on the packet. In the first case, a node could be selfish and not want to waste power with retransmissions. In the latter case, the only reason B would have for taking the actions that it does is because it is malicious. B wastes battery power and CPU time, so it is not selfish. An overloaded node would not engage in this behavior either, since it wastes badly needed CPU time and bandwidth. Thus, this second case should be a rare occurrence.

Another problem can occur when nodes falsely report other nodes as misbehaving. A malicious node could attempt to partition the network by claiming that some nodes following it in the path are misbehaving. For instance, node A could report that node B is not forwarding packets when in fact it is. This will cause S to mark B as misbehaving when A is the culprit. This behavior, however, will be detected. Since A is passing messages on to B (as verified by S), then any acknowledgements from D to S will go through A to S, and S will wonder why it receives replies from D when supposedly B dropped packets in the forward direction. In addition, if A drops acknowledgements to hide them from S, then node B will detect this misbehavior and will report it to D.

Another problem is that a misbehaving node that can control its transmission power can circumvent the watchdog. A node could limit its transmission power such that the signal is strong enough to be overheard by the previous node but too weak to be received by the true recipient. This would require that the misbehaving node know the transmission power required to reach each of its neighboring nodes. Only a node with malicious intent would behave in this manner — selfish nodes have nothing to gain since battery power is wasted and overloaded nodes would not relieve any congestion by doing this.

Multiple nodes in collusion can mount a more sophisticated attack. For example, B and C from Figure 2 could collude to cause mischief. In this case, B forwards a packet to C but does not report to A when C drops the packet. Because of this limitation, it may be necessary to disallow two consecutive untrusted nodes in a routing path. In this paper, we only deal with the possibility of nodes acting alone. The harder problem of colluding nodes is being studied by Johnson at CMU [13].

Finally, a node can circumvent the watchdog by dropping packets at a lower rate than the watchdog's configured minimum misbehavior threshold. Although the watchdog will not detect this node as misbehaving, this node is forced to forward at the threshold bandwidth. In this way the watchdog serves to enforce this minimum bandwidth.

The watchdog mechanism could be used to some degree to detect replay attacks but would require maintaining a great deal of state information at each node as it monitors its neighbors to ensure that they do not retransmit a packet that they have already forwarded. Also, if a collision has taken place at the receiving node, it would be neccesary and correct for a node to retransmit a packet, which may appear as a replay attack to the node acting as its watchdog. Therefore, detecting replay attacks would neither be an efficient nor an effective use of the watchdog mechanism.

For the watchdog to work properly, it must know where a packet should be in two hops. In our implementation, the watchdog has this information because DSR is a source routing protocol. If the watchdog does not have this information (for instance if it were implemented on top of a hop-by-hop routing protocol), then a malicious or broken node could broadcast the packet to a non-existant node and the watchdog would have no way of knowing. Because of this limitation, the watchdog works best on top of a source routing protocol.

3.2 Pathrater

The pathrater, run by each node in the network, combines knowledge of misbehaving nodes with link reliability data to pick the route most likely to be reliable. Each node maintains a rating for every other node it knows about in the network. It calculates a path metric by averaging the node ratings in the path. We choose this metric because it gives a comparison of the overall reliability of different paths and allows pathrater to emulate the shortest length path algorithm when no reliability information has been collected, as explained below. If there are multiple paths to the same destination, we choose the path with the highest metric. Note that this differs from standard DSR, which chooses the shortest path in the route cache. Further note that since the pathrater depends on knowing the exact path a packet has traversed, it must be implemented on top of a source routing protocol.

The pathrater assigns ratings to nodes according to the following algorithm. When a node in the network becomes known to the pathrater (through route discovery), the pathrater assigns it a "neutral" rating of 0.5. A node always rates itself with a 1.0. This ensures that when calculating path rates, if all other nodes are neutral nodes (rather than suspected misbehaving nodes), the pathrater picks the shortest length path. The pathrater increments the ratings of nodes on all actively used paths by 0.01 at periodic intervals of 200 ms. An actively used path is one on which the node has sent a packet within the previous rate increment interval. The maximum value a neutral node can attain is 0.8. We decrement a node's rating by 0.05 when we detect a link break during packet forwarding and the node becomes unreachable. The lower bound rating of a "neutral" node is 0.0. The pathrater does not modify the ratings of nodes that are not currently in active use.

We assign a special highly negative value, -100 in the simulations, to nodes suspected of misbehaving by the watchdog mechanism. When the pathrater calculates the path metric, negative path values indicate the existence of one or more suspected misbehaving nodes in the path. If a node is marked as misbehaving due to a temporary malfunction or incorrect accusation it would be preferrable if it were not permanently excluded from routing. Therefore nodes that have negative ratings should have their ratings slowly increased or set back to a non-negative value after a long timeout. This is not implemented in our simulations since the current simulation period is too short to reset a misbehaving node's rating. Section 5.3 discusses the effect on throughput of accusing well-behaving nodes.

When the pathrater learns that a node on a path that is in use misbehaves, and it cannot find a path free of misbehaving nodes, it sends out a ROUTE REQUEST if we have enabled an extension we call Send Route Request (SRR).

4. METHODOLOGY

In this section we describe our simulator, simulation parameters, and measured metrics.

We use a version of Berkeley's Network Simulator (ns) [8] that includes wireless extensions made by the CMU Monarch project. We also use a visualization tool from CMU called ad-hockey [25] to view the results of our simulations and detect overall trends in the network. To execute the simulations, we use PCs (450 or 500 MHz Pentium IIIs with at least 128 MB of RAM) running Red Hat Linux 6.1.

Our simulations take place in a 670 by 670 meter flat space filled with a scattering of 50 wireless nodes. The physical layer and the 802.11 MAC layer we use are included in the CMU wireless extensions to ns[3].

4.1 Movement and Communication Patterns

The nodes communicate using 10 constant bit rate (CBR) node-to-node connections. Four nodes are sources for two connections each, and two nodes are sources for one connection each. Eight of the flow destinations receive only one flow and the ninth destination receives two flows. The communication pattern we use was developed by CMU [3].

In all of our node movement scenarios, the nodes choose a destination and move in a straight line towards the destination at a speed uniformly distributed between 0 meters/second (m/s) and some maximum speed. This is called the random waypoint model [3]. We limit the maximum speed of a node to 20 m/s (10 m/s on average) and we set the run-time of the simulations to 200 seconds. Once the node reaches its destination, it waits for the *pause time* before choosing a random destination and repeating the process. We use pause times of 0 and 60 seconds. In addition we use two different variations of the initial node placement and movement patterns. By combining the two pause times with two movement patterns, we obtain four different mobility scenarios.

4.2 Misbehaving Nodes

Of the 50 nodes in the simulated network, some variable percentage of the nodes misbehave. In our simulations, a misbehaving node is one that agrees to participate in forwarding packets (it appends its address into ROUTE REQUEST packets) but then indiscriminately drops all data packets that are routed through it.

We vary the percentage of the network comprised of misbehaving nodes from 0% to 40% in 5% increments. While a network with 40% misbehaving nodes may seem unrealistic, it is interesting to study the behavior of the algorithms in a more hostile environment than we hope to encounter in real life. We use Tcl's [17] built-in pseudo-random number generator to designate misbehaving nodes randomly. We use the same seed across the 0% to 40% variation of the misbehaving nodes parameter, which means that the group of misbehaving nodes in the 10% case is a superset of the group of misbehaving nodes in the 5% case. This ensures that the obstacles present in lower percentage misbehaving node runs are also present in higher percentage misbehaving node runs.

4.3 Metrics

We evaluate our extensions using the following three metrics:

- Throughput: This is the percentage of sent data packets actually received by the intended destinations.
- Overhead: This is the ratio of routing-related transmissions (ROUTE REQUEST, ROUTE REPLY, ROUTE ERROR, and watchdog) to data transmissions in a simulation. A transmission is one node either sending or forwarding a packet. For example, one packet being forwarded across 10 nodes would count as 10 transmissions. We count transmissions instead of packets because we want to compare routing-related transmissions to data transmissions, but some routing packets are more expensive to the network than other packets: ROUTE REQUEST packets are broadcast to all neighbors which in turn broadcast to all of their neighbors, causing a tree of packet transmissions. Unicast ROUTE REPLY, ROUTE ERROR, watchdog, and data packets only travel along a single path.
- Effects of watchdog *false positives* on network throughput. False positives occur when the watchdog mechanism reports that a node is misbehaving when in fact it is not, for reasons discussed in Section 3. We study the impact of this on throughput.

5. SIMULATION RESULTS

In this section we present the results of our simulations. We focus on three metrics of evaluation: network throughput, routing overhead, and the effects of false positives on throughput.

We test the utility of various combinations of our extensions: watchdog (WD), pathrater (PR), and send (extra) route request (SRR). We use the SRR extension to find new paths when all known paths include a suspected misbehaving node. Each of the following sections includes two graphs of simulation results for two separate pause times. The first graph is for a pause time of 0 (the nodes are in constant motion) and the second is for a pause time of 60 seconds before and in between node movement. We simulate two different node mobility patterns using four different pseudo-random number generator seeds. The seeds determine which nodes misbehave. We plot the average of the eight simulations.

	Maximum	Minimum
0 second pause time	88.6%	75.2%
60 second pause time	95.0%	73.9%

Table 1: Maximum and minimum network throughput obtained by any simulation at 40% misbehaving nodes with all features enabled.

5.1 Network Throughput

We graph four curves for network throughput: everything enabled, watchdog and pathrater enabled, only pathrater enabled, and everything disabled. We choose to graph both everything enabled and everything enabled except SRR, because we want to isolate performance gains or problems caused by extra route requests. Since the pathrater is not strictly a tool to be used for circumventing misbehaving nodes, we choose to include the graph where only pathrater is enabled to determine if it increases network throughput without any knowledge of suspected misbehaving nodes. We do not graph watchdog and SRR activated without pathrater, since without pathrater the information about misbehaving nodes would not be used for routing decisions.

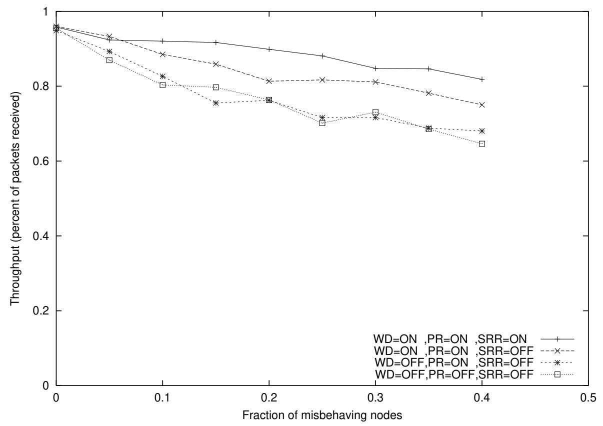
Figure 5 shows the total network throughput, calculated as the fraction of data packets generated that are received, versus the fraction of misbehaving nodes in the network for the combinations of extensions. In the case where the network contains no misbehaving nodes, all four curves achieve around 95% throughput. After the 0% misbehaving node case, the graphs diverge.

As expected, the simulations with all three extensions active perform the best by a considerable margin as misbehaving nodes are added to the network. The mechanisms increase the throughput by up to 27% compared to the basic protocol, maintaining a throughput greater than 80% for both pause times, even with 40% misbehaving nodes. Table 1 lists the maximum and minumum throuput achieved in any simulation run at 40% misbehaving nodes with all options enabled.

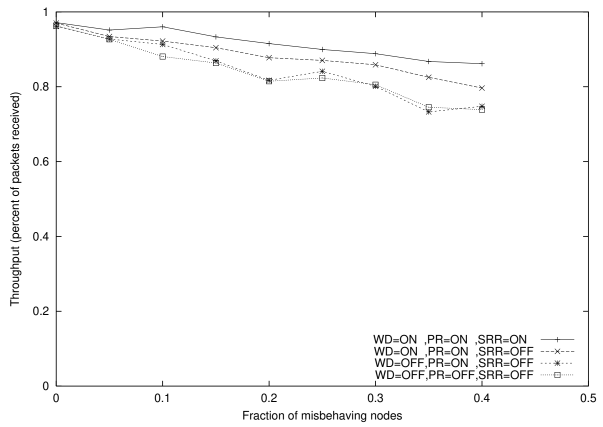
When a subset of the extensions is active, performance does not increase as much over the simulations with no extensions. Watchdog alone does not affect routing decisions, but it supplies pathrater with extra information to combat misbehaving nodes more effectively. When watchdog is deactivated,

	Maximum	Minimum
0 second pause time	31.3%	18.9%
60 second pause time	23.5%	11.0%

Table 2: Maximum and minimum overhead obtained by any simulation at 40% misbehaving nodes with all features enabled.



(a) 0 second pause time



(b) 60 second pause time

Figure 5: Overall network throughput as a function of the fraction of misbehaving nodes in the network.

the source node has no way of detecting the misbehaving node in its path to the destination, and so its transmission flow suffers total packet loss. Pathrater alone cannot detect a path with misbehaving nodes to decrement its rate (see Section 7).

One effect of the randomness of ns is that nodes may receive route replies to their route requests in a different order in one simulation than in another simulation with slightly varied parameters. This change can result in a node choosing a path with a misbehaving node in one run, but not choosing that path in a simulation with more misbehaving nodes in the network. This may actually result in slight increases in network throughput when the number of misbehaving nodes increases. For instance, this is noticeable in the pathrater-only curve of Figure 5 (b) where the throughput raises from 82% to 84% between 20% and 25% misbehaving nodes.

In both throughput graphs, the everything disabled curve and the pathrater only curves closely follow each other. From the graphs we conclude that the pathrater alone does not significantly affect performance. In Section 7 we suggest some improvements to the pathrater that may increase its utility in the absence of the other extensions.

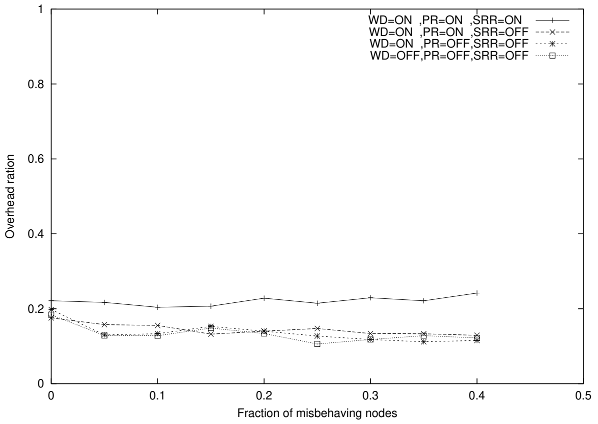
5.2 Routing Overhead

For routing overhead, we graph four curves: everything on, pathrater and watchdog on, only watchdog on (watchdog-only), and everything off. Using the everything off graph as our basis for comparison, we graph the watchdog-only curve to find the overhead generated just by the watchdog when it sends notifications to senders. The watchdog and pathrater curve shows the overhead added by watchdog and pathrater but with pathrater's ability to send out extra route requests disabled. The everything on curve includes the overhead created by pathrater when sending out extra route requests.

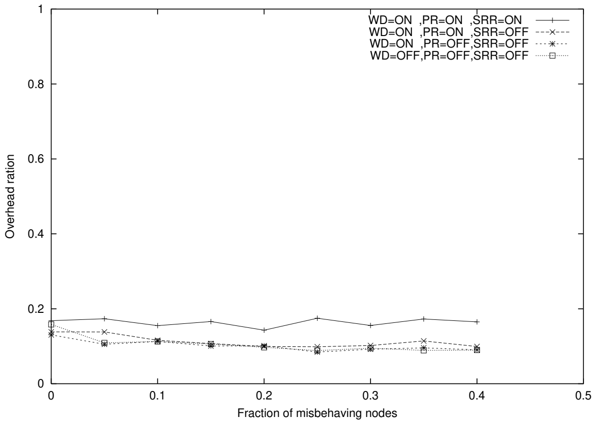
Figure 6 shows the amount of overhead incurred by activating the different routing extensions. The greatest effect on routing overhead results from using the SRR feature, which sends out route requests for a destination to which the only known routes include suspected misbehaving nodes. For 40% misbehaving nodes in the high mobility scenario, the overhead rises from 12% to 24% when SRR is activated in the pathrater. Any route requests generated by SRR will flood the network with ROUTE REQUEST and ROUTE REPLY packets, which greatly increase the overhead. Table 2 lists the maximum and minimum overhead for any of the simulations with all options enabled at 40% misbehaving nodes.

The watchdog mechanism itself only adds a very small amount of extra overhead as seen by comparing the watchdog-only graph with the all-disabled graph. Also, the added overhead is not affected by the increase in misbehaving nodes in the network. Using both the watchdog and pathrater mechanisms increases the throughput of the network by 16% at 40% misbehaving nodes with only 6% additional network overhead (see Figure 6 (a)).

Though the overhead added by these extensions is significant, especially when pathrater sends out route requests to avoid misbehaving nodes, these extensions still improve net throughput. Therefore, the main concerns with high overhead involve issues such as increased battery usage on portables and PDAs. Since the largest factor accounting for the overhead is route requests, the overhead can be significantly reduced by optimizing the delay between pathrater sending out route requests and incorporating some of the approaches developed for mitigating route requests and broadcast storms in general [1, 4, 14].



(a) 0 second pause time



(b) 60 second pause time

Figure 6: This figure shows routing overhead as a ratio of routing packet transmissions to data packet transmissions. This ratio is plotted against the fraction of misbehaving nodes.

5.3 Effects of False Detection

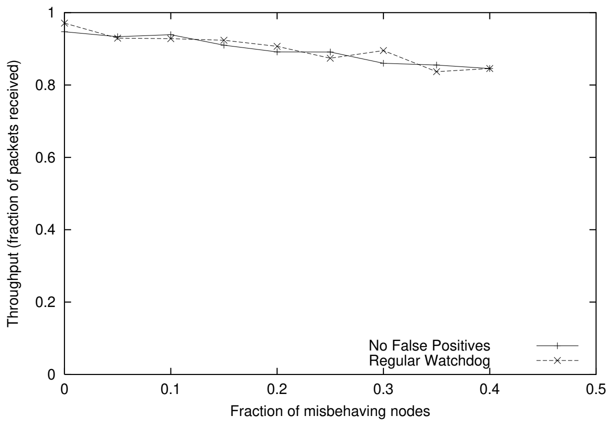
We compare simulations of the regular watchdog with a watchdog that does not report false positives. Figure 7 shows the network throughput lost by the watchdog incorrectly reporting well-behaved nodes. These results show that throughput is not appreciably affected by false positives and that they may even have beneficial side effects, as described below.

The similarity in throughput can be attributed to a few factors. First, the nodes incorrectly reported as misbehaving could have moved out of the previous node's listening range before forwarding on a packet. If these nodes move out of range frequently enough to warrant an accusation of misbehavior they may be unreliable due to their location, and the source would be better off routing around them. The fact that more false positives are reported in the 0 second pause time simulations as compared to the 60 second pause time simulations, as shown in Table 3, supports this conclusion. Table 3 shows the average value of false positives reported by the simulation runs for each pause time and misbehaving node percentage.

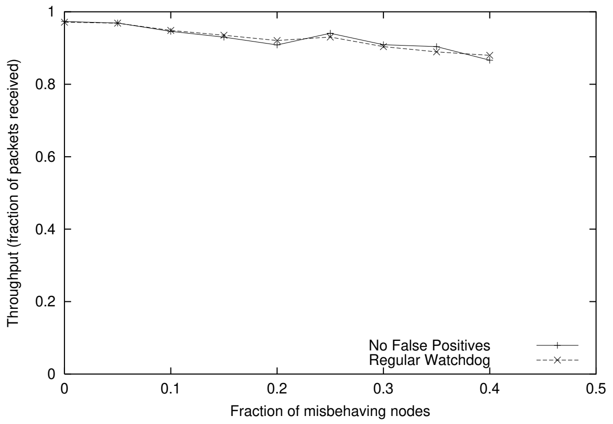
Another factor that may account for the similar throughput of the watchdog's performance with and without false positives concerns one of the limitations of the watchdog. As described in Section 3, if a collision occurs while the watchdog is waiting for the next node to forward a packet, it may never overhear the packet being transmitted. If many collisions occur over time, the watchdog may incorrectly assume that the next node is misbehaving. However, if a node constantly experiences collisions, it may actually increase throughput to route packets around areas of high communication density.

Percent misbehaving nodes	0%	5%	10%	15%	20%	25%	30%	35%	40%
0 second pause time	111.2	82.8	90.3	66.5	75.5	60.8	67.5	31.3	50.8
60 second pause time	39.0	57.6	40.8	63.1	35.7	79.5	46.7	21.7	47.2

Table 3: Comparison of the number of false positives between the 0 second and 60 second pause time simulations. Average taken from the simulations with all features enabled.



(a) 0 second pause time



(b) 60 second pause time

Figure 7: Comparison of network throughput between the regular watchdog and a watchdog that reports no false positives.

Yet another factor is that increased false positives will result in more paths including a suspected misbehaving node. The pathrater will then send out more route requests to the destination. This increases the overhead in the network, but it also provides the sending node with a fresher list of routes for its route cache.

6. RELATED WORK

To our knowledge, there is no previously published work on detection of routing misbehavior specific to ad hoc networks, although there is relevant work by Smith, Murthy and Garcia-Luna-Aceves on securing distance vector routing protocols from Byzantine routing failures [22]. In their work, they suggest countermeasures to secure routing messages and routing updates. This work may be applicable to ad hoc networks in that distance vector routing protocols, such as DSDV, have been proposed for ad hoc networks.

Zhou and Haas investigate distributed certificate authorities in ad hoc networks using threshold cryptography[27]. Zhou and Haas take the view that no one single node in an ad hoc network can be trusted due to low physical security and low availability. Therefore, using a single node to provide an important network-wide service, such as a certificate authority, is very risky. Threshold cryptography allows a certificate authority's private key to be broken up into shares and distributed across multiple nodes. To sign a certificate, a subset of the nodes with private key shares must jointly collaborate. Thus, to mount a successful attack on the certificate authority, an intruder must compromise multiple nodes.

To further frustrate attack attempts over time, Zhou and Haas' scheme uses share refreshing. It is possible that over a long period of time enough share servers could be compromised to recover the certificate authority's secret key. Share refreshing allows uncompromised servers to compute a new private key periodically from the old private key's shares. This periodic refreshing means that an attacker must infiltrate a large number of nodes within a short time span to recover the certificate authority's secret key.

Stajano and Anderson [23] elucidate some of the security issues facing ad hoc networks and investigate ad hoc networks composed of low compute-power nodes such as home appliances, sensor networks, and PDAs where full public key cryptography may not be feasible. The authors develop a system in which a wireless device "imprints" itself on a master device, accepting a symmetric encryption key from the first device that sends it a key. After receiving that key, the slave device will not recognize any other device as a master except the device that originally sent it the key. The authors bring up an interesting denial of service attack: the battery drain attack. A misbehaving node can mount a denial-of-service attack against another node by routing seemingly legitimate traffic through the node in an attempt to wear down the other node's batteries.

7. FUTURE WORK

This paper presents initial work in detecting misbehaving nodes and mitigating their performance impact in ad hoc wireless networks. In this section we describe some further ideas we would like to explore.

We plan on conducting more rigorous tests of the watchdog and pathrater parameters to determine optimal values to increase throughput in different situations. Currently we are experimenting with different watchdog thresholds for deciding when a node is misbehaving. Some of the variables to optimize for the pathrater include the rating increment and decrement amounts, the rate incrementing interval, and the delay between sending out route requests to decrease the overhead caused by this feature.

Our simulations use scenarios in which there are no *a priori* trust relationships, but we expect the performance of pathrater to increase when it can make use of explicitly trusted nodes. Trusted node lists are available in some ad hoc network scenarios, and we would like to analyze the performance of our routing extensions in these scenarios.

Currently the pathrater only decrements a node's rating when another node tries unsuccessfully to send to it or if the watchdog mechanism is active and determines that a node is misbehaving. Without the watchdog active, the pathrater cannot detect misbehaving nodes. An obvious enhancement would be to receive updates from a reliable transport layer, such as TCP, when ACKs fail to be received. This would allow the pathrater to detect bad paths and lower the nodes' ratings accordingly.

All the simulations presented in this paper use CBR data sources with no reliability requirements. Our next goal is to analyze how the routing extensions perform with TCP flows common to most network applications. Our focus would then change from measuring throughput, or dropped packets, to measuring the time to complete a reliable transmission, such as an FTP transfer. For these tests the modification to pathrater described above should improve performance significantly in the case where the watchdog is not active.

Finally, we would like to evaluate the watchdog and pathrater considering latency in addition to throughput.

8. CONCLUSION

Ad hoc networks are an increasingly promising area of research with practical applications, but they are vulnerable in many settings to nodes that misbehave when routing packets. For robust performance in an untrusted environment, it is necessary to resist such routing misbehavior.

In this paper we analyze two possible extensions to DSR to mitigate the effects of routing misbehavior in ad hoc networks - the watchdog and the pathrater. We show that the two techniques increase throughput by 17% in a network with moderate mobility, while increasing the ratio of overhead transmissions to data transmissions from the standard routing protocol's 9% to 17%. During extreme mobility, watchdog and pathrater can increase network throughput by 27%, while increasing the percentage of overhead transmissions from 12% to 24%.

These results show that we can gain the benefits of an increased number of routing nodes while minimizing the effects of misbehaving nodes. In addition we show that this can be done without *a priori* trust or excessive overhead.

9. ACKNOWLEDGEMENTS

We would like to thank Diane Tang, Petros Maniatis, Mema Roussopoulos, and Ed Swierk for their comments on drafts of this paper. We would also like to thank Dan Boneh for his help in early discussions of this work. This work was supported in part by a generous gift from NTT Mobile Communications Network, Inc. (NTT DoCoMo). In addition, Sergio Marti was supported by a National Defense Science and Engineering Graduate Fellowship.

10. REFERENCES

[1] S. Basagni and et al. A Distance Routing Effect Algorithm for Mobility (DREAM). In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, October 1998.

[2] V. Bharghavan, A. Demers, S. Shenker, and L. Zhang. MACAW: A Medium Access Protocol for Wireless LANs. *In Proceedings of ACM SIGCOMM '94*, August 1994.

[3] J. Broch, D. A. Maltz, D. B. Johnson, Y. C. Hu, and J. Jetcheva. A Performance Comparison of Multi-Hop Wireless Ad Hoc Network Routing Protocols. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, October 1998.

[4] R. Castaneda and S. R. Das. Query Localization Techniques for On-Demand Routing Protocols in Ad Hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, August 1999.

[5] S. Corson and V. Park. Temporally-Ordered Routing Algorithm (TORA) Version 1 Functional Specification. *Mobile Ad-hoc Network (MANET) Working Group, IETF*, October 1999.

[6] B. P. Crow, I. K. Widjaja, G. Jeong, and P. T. Sakai. IEEE-802.11 Wireless local Area Networks. *IEEE Communications Magazine*, vol. 35, No.9: pages 116-126, September 1997.

[7] S. Das, C. E. Perkins and E. M. Royer. Ad Hoc On Demand Distance Vector (AODV) Routing (Internet-Draft). *Mobile Ad-hoc Network (MANET) Working Group, IETF*, October 1999.

[8] K. Fall and K. Varadhan, editors. ns notes and documentation. The VINT Project, UC Berkeley, LBL, USC/ISI, and Xerox PARC, July 1999. Available from `http://www-mash.cs.berkeley.edu/ns/`.

[9] J.J. Garcia-Luna-Aceves and M. Spohn. Source-Tree Routing in Wireless Networks. In *Proceedings IEEE ICNP 99: 7th International Conference on Network Protocols*, Toronto, Canada, October 31–November 3, 1999.

[10] J.J. Garcia-Luna-Aceves, Marcelo Spohn, and David Beyer. Source Tree Adaptive Routing (STAR) Protocol (Internet-Draft). *Mobile Ad hoc Network (MANET) Working Group, IETF,* October 1999.

[11] P. Johansson and T. Larsson. Scenario-Based Performance Analysis of Routing Protocols for Mobile Ad-Hoc Networks. In *Proceedings of the Fifth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '99)*, August 1999.

[12] D. Johnson, D. A. Maltz, and J. Broch. The Dynamic Source Routing Protocol for Mobile Ad Hoc Networks (Internet-Draft). *Mobile Ad-hoc Network (MANET) Working Group, IETF*, October 1999.

D. Johnson. Personal Communication. February 2000.

[13] J. Jubin and J. Tornow. The DARPA Packet Radio Network Protocols. In *Proceedings of the IEEE*, 75(1):21-32, 1987.

[14] Y.-B. Ko and N. H. Vaidya. Location-Aided Routing (LAR) in Mobile Ad Hoc Networks. In *Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MOBICOM '98)*, October 1998.

[15] Y.-B. Ko and N. H. Vaidya. Geocasting in Mobile Ad Hoc Networks: Location-Based Multicast Algorithms. *WMCSA'99*, New Orleans.

[16] IETF MANET Working Group Internet Drafts. `http://www.ietf.org/ids.by.wg/manet.html`.

[17] J. K. Ousterhout. *Tcl and the Tk Toolkit.* Addison-Wesley, 1994.

[18] S. H. Park, A. Ganz, and Z. Ganz. Security protocol for IEEE 802.11 wireless local area network. *Mobile Networks and Applications*. Vol. 3. 1998.

[19] C.E. Perkins and P. Bhagwat. Highly Dynamic Destination-Sequenced Distance-Vector Routing (DSDV) for Mobile Computers. In *Proceedings of the SIGCOMM '94 Conference on Communications Architectures, Protocols and Applications*, pages 234-244, August 1994.

[20] R. Prakash. Unidirectional Links Prove Costly in Wireless Ad Hoc Networks. In *Proceedings of DIMACS Workshop on Mobile Networks and Computers*, 1999.

[21] B. Smith and J.J. Garcia-Luna-Aceves. Efficient Security Mechanisms for the Border Gateway Routing Protocol. *Computer Communications (Elsevier)*, Vol. 21, No. 3: pp. 203-210, 1998, .

[22] B. Smith, S. Murthy, and J.J. Garcia-Luna-Aceves. Securing Distance-Vector Routing Protocols. In *Proceedings of Internet Society Symposium on Network and Distributed System Security*, San Diego, CA, February 1997.

[23] F. Stajano and R. Anderson. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. B. Christianson, B. Crispo, and M. Roe (Eds.)., *Security Protocols, 7th International Workshop Proceedings, Lecture Notes in Computer Science*, 1999.

[24] D. G. Stinson. *Cryptography: Theory and Practice.* CRC Press, 1995.

[25] The CMU Monarch Project. The CMU Monarch Projects Wireless and Mobility Extensions to ns. `http://www.monarch.cs.cmu.edu/cmu-ns.html`. Oct. 12, 1999.

[26] C.-K. Toh. Associativity Based Routing For Ad-Hoc Mobile Networks. *Wireless Personal Communications Journal, Special Issue on Mobile Networking and Computing Systems*, Vol. 4, No. 2, pp.103-139, March 1997.

[27] L. Zhou and Z. J. Haas. Securing Ad Hoc Networks. *IEEE Network Magazine*, vol. 13, no.6, November/December 1999.