The AD-MIX Protocol for Encouraging Participation in Mobile Ad hoc Networks

Swaminathan Sundaramurthy Elizabeth M. Belding-Royer
Department of Computer Science
University of California, Santa Barbara
{swami, ebelding}@cs.ucsb.edu

Abstract

Mobile ad hoc networks are autonomous self-organized networks in which each node relies on the other nodes in the network to perform routing on its behalf. Proper functioning of the network is dependent on participation and cooperation of the nodes in routing and packet forwarding. Unfortunately, providing these services may not be in the best interest of a mobile node, since it results in the depletion of the node's resources. Selfish behavior by a node may result in degraded network performance due to denial of service, decrease in network throughput and partitioning of the network. Because it is in a node's interest to not forward traffic, nodes should be given some form of incentive for the services they provide.

In this paper, we address the problem of selfishness in mobile ad hoc networks by proposing a protocol called AD-MIX that encourages participation. AD-MIX discourages selfishness by concealing the true destination of packets from intermediate nodes along the path, forcing a node to participate or risk dropping packets destined for itself. Simulation results show that employing AD-MIX encourages participation without a significant increase in overhead. In addition to encouraging participation, AD-MIX also facilitates anonymization and secure communication between nodes.

1 Introduction

Mobile ad hoc networks are autonomous, self-organized networks. In these networks, services such as packet forwarding, routing and mobility management are handled by each device on behalf of the network. In military and emergency situations, nodes are usually controlled by a single authority and work toward a common goal; however, in civilian networks each node joins the network for its own benefit. Nodes under individual control may not trust each other and have no incentive to forward packets destined for other nodes.

An ad hoc network composed of mobile nodes has the following properties:

- Providing services results in the depletion of the provider's resources.
- The decision to provide services is left to the discretion of each device.

Mobile devices typically have constrained resources such as battery life, CPU cycles, and available network bandwidth; providing services to the other nodes in the network further reduces the node's resources. Therefore it can be appealing for a node in a non-cooperative network to behave selfishly to preserve its resources.

It has been shown that the presence of selfish nodes can reduce the throughput of the network drastically. This may result in partitioning or, under extreme circumstances, complete breakdown of the network. According to a previous study [12], when 10% - 40% of the nodes in the network misbehave, the average throughput may degrade by 16% - 32%.

Selfishness is therefore a serious concern. Since it is not in the interest of mobile devices to provide services to other nodes, incentives of some form should be provided to encourage nodes to participate. This paper presents AD-MIX, a protocol that encourages participation of nodes by concealing the true destination of the packet. This discourages a selfish node from dropping packets, since those packets may be ultimately destined for itself. The protocol can be implemented as a micro-layer above the network later of the OSI protocol stack.

The remainder of the paper is organized as follows. Related work is discussed in Section 2. Section 3 identifies the requirements of an approach that guarantees participation and enumerates the assumptions made by AD-MIX. The operation of the AD-MIX protocol is introduced in Section 4, and its performance is evaluated in Section 5. A short discussion about AD-MIX is presented in Section 6. Finally, our conclusions and future work are presented in Section 7.

2 Related Work

Numerous approaches have been suggested to ensure participation in ad hoc networks. Buttyán and

Hubaux [5] suggest the use of *Nuglets* as a virtual currency that serves as per-hop per-packet payment to stimulate co-operation in ad hoc networks. Nodes are charged for services they use and rewarded for providing services. Thus nodes are encouraged to forward packets to earn nuglets to use the services of the network. Some of the drawbacks of the approach include: the sources are charged for dropped packets, estimation of the cost of transmitting a packet is non-trivial, and no feedback mechanism exists to alleviate problems of over-estimation. Also, since the nodes closer to the center of the network tend to handle more packets, the distribution of nuglets is greater among these nodes. Thus the peripheral nodes spend more than they can earn, resulting in *bankruptcy* and eventually denial of service.

The *Watchdog* protocol [12] requires that the nodes detect non-forwarding nodes through passive listening and report these nodes to the packet source. The *Pathrater* [12] protocol at the source node rates paths based on the information it has about the nodes along the path to the destination to choose the most reliable path for subsequent packets. One of the drawbacks of the protocol is that it is possible for a malicious node to send erroneous information degrading the ratings of certain paths. Also, the protocol does not punish selfish nodes by blocking packets from them; it simply routes packets around them. This approach potentially encourages selfishness, because if a node behaves maliciously, then fewer packets are transmitted through it; however, the misbehaving node still receives all packets destined for it.

In [3], the authors propose a protocol called CONFIDANT where each node monitors its neighbors' behavior and maintains a reputation for each neighbor based on its behavior. This reputation is propagated to the other nodes in the network to inform them of misbehaving nodes. This is similar to [12], except that in CONFIDANT, nodes maintain the reputation of other nodes instead of paths. Since there is no mechanism to verify the authenticity of the misbehavior notifications received, malicious nodes may send false notifications to implicate innocent nodes.

As discussed, there are several drawbacks in the existing economic [5] and reputation-based models [3, 12]. AD-MIX circumvents the shortcomings of economic models by using non-economic incentives. Further, AD-MIX is also immune to the drawbacks of the reputation-based models since intermediate nodes do not perform any important functions that can be exploited by malicious nodes. In the following section, we identify the characteristics that a protocol providing participation incentives should possess.

3 Problem formulation

A good incentive system should possess the following characteristics:

- *Inviolable:* Should not be possible to violate or circumvent the incentive system.
- *Decentralized:* The approach should preserve the distributed nature, since centralized services can cause scalability problems.
- *Low Overhead:* The approach should introduce minimal control and data overhead to avoid degradation of the performance of the network.
- *Universal:* The approach should be universal in the sense that it should not impose specialized hardware or architecture requirements on the participating nodes.
- *Fair:* The scheme should be fair to all nodes.
- *Stable:* The incentive should not degrade with time.

The characteristics stated above serve as our design goals. With these design goals in mind, we now briefly describe the assumptions made by our protocol.

- *Nodes may be selfish but not malicious*
  As with most economic models [4, 5], AD-MIX assumes that the network contains only *selfish* nodes. It does not handle the presence of *malicious* nodes. Selfish nodes may be distinguished from malicious nodes as follows:

  Selfish nodes are nodes that utilize services provided by others but do not reciprocate [5]. The primary motivation of these nodes is to preserve their resources. These nodes do not have harmful intentions toward the network, though their actions may adversely affect the performance of the network.

  Malicious nodes are nodes that join the network with the intent of harming it by causing network partitions, denial of service, etc.

  AD-MIX assumes the presence of an infrastructure to detect and handle malicious nodes. A number of security protocols have been proposed to handle malicious nodes [2, 12].

- *Asymmetric key distribution infrastructure*
  Each node is assumed to possess a pair of unique asymmetric keys. In addition, AD-MIX requires each node to possess the public keys of all the nodes it wishes to route through. Hence, an infrastructure for key distribution is necessary. Detailed explanations of the various key distribution mechanisms for ad hoc networks are discussed in [1, 6, 11].

4 Description of the AD-MIX Protocol

In the previous section, we discussed the design goals and assumptions made by AD-MIX. In this section we describe the operation of AD-MIX. Before describing the protocol, some background information and an analogy to aid in understanding the protocol are presented.

4.1 Background

The functioning of the AD-MIX protocol is similar in principle to the Mixnet [7] approach. Mixnet provides anonymity of message sources in wired networks. AD-MIX adapts the technique employed by Mixnets to encourage participation in wireless ad hoc networks.

In the Mixnet protocol, a mix is a proxy computer within a mix network, or a mix cascade, that collects the encoded messages sent by the various users of a communications network. It re-encodes these messages and changes their order before transmitting them to the next mix in line. At the next mix, this procedure is repeated. The last mix along the route identifies the actual receiver and delivers the encoded message to that node. Therefore, a mix network guarantees effective anonymization, even in the presence of just one reliable mix.

AD-MIX is an adaptation of this technique that promotes participation through information hiding, i.e., none of the nodes through which the packets pass are aware of the ultimate destination of the packet. The destinations of the packets are obscured by using nodes, called *poles*, whose function is similar to the mix servers of the Mixnet protocol. Since mixes are used to conceal the true destination of the packets, selfish nodes cannot risk dropping a packet based on the packet's destination address. By dropping packets, they may miss packets potentially destined for them. Therefore, nodes are encouraged to forward all packets passing through them.

In the Mixnet protocol, there is no limit on the number of mix servers used; however, with each additional mix server, the path length of the packet increases resulting in increased packet delivery time. In ad hoc networks, lack of routes to poles may result in control message floods. Hence the number of poles chosen should be minimal. In Section 4.5 we prove that two poles are sufficient to guarantee participation. In addition, by choosing a variable number (between 0 and 2) of poles, the average number of poles employed per packet is reduced. This results in reduced average path length and delivery times.

We now present an analogy to convey the basic operation of AD-MIX and demonstrate how it encourages participation.

4.2 Chests and Lazy People

Consider a group of people (nodes) who wish to communicate with each other. A person interested in communicating writes out the message in a letter (packet) with his name and the name of the intended recipient and hands it to his neighbor. This neighbor in turn forwards this to his neighbor. This process continues until the message reaches the intended recipient.

This method of communication would be sufficient if everyone in the group were well-behaved. Let us now assume that the group also has some lazy people (selfish nodes). Lazy people are not eager to forward letters destined for other people since they do not benefit from it. Hence, there is a possibility of them not forwarding (dropping) the message. In order to ensure that everyone in the group participates in forwarding messages, it is sufficient to coerce the lazy people to forward messages.

To address the presence of lazy people, we employ a different protocol to send messages. We assume the presence of chests (safe-boxes) that can be opened or closed with exactly two keys – a private key and a public key. These keys cannot be duplicated, and a chest locked with one key can only be opened by its pair. We also assume that each person has a unique private key and that the sender knows the public key of each recipient to whom he wants to transmit messages.

When a person wishes to communicate, he secures the message inside several nested chests before sending it. The number of chests chosen may vary for each message, and is decided by the sender. Each chest is locked with the public key of a member of the group and can be opened only by the person who possesses the private key. Each chest is also labeled with the name of the person who possesses the private key to open it. The innermost chest is always locked with the public key of the ultimate message recipient.

A person interested in conveying a message 'hides' the message in one or more chests and forwards it to the neighbor closest to the outermost recipient. On receiving a chest addressed to him, the person opens it with his private key. If he finds a message inside, then he is the intended recipient of the message, so he keeps it. On the other hand, if he finds another chest inside it, he forwards it to his neighbor closest to the recipient of the inner chest. It is impossible for any person forwarding the chest to tell if it is the innermost chest, unless he can open it to find the message inside it.

This scheme causes the true destination of the message to be concealed from the people who handle the chest. A lazy person, on receiving a chest, will not be able drop it, since there exists a possibility of it housing another chest containing a message

intended for him, which he can only get after the outer chests have been opened by their respective recipients. Choosing not to forward the chest may cause him to drop a message destined for himself. Hence he is forced to forward chests destined for other recipients. In order to increase the probability of a selfish person dropping his own message, the paths of some of the chests are chosen so that the chest loops back to a person who had previously forwarded it. This can be achieved by addressing an outer chest to a person, where the actual receiver is along the path to that person. A lazy person would drop messages destined for him if he behaved selfishly. Thus, the lazy people in the group are also encouraged to co-operate in message forwarding.

4.3 Foundations of the AD-MIX Protocol

With respect to a packet, a node can be classified as a source node, destination node, polar node or non-polar node. Polar nodes, also called poles, are synonymous to the mix servers of the Mixnet protocol. The operation of AD-MIX can be explained by describing its operation at the source, non-polar and polar nodes.

4.3.1 Operation at the Source Node

When a node wants to transmit a packet, it chooses between 0 and 2 nodes from the network through which the packets should pass. Poles can either be randomly picked, or for better performance chosen as explained in Section 4.4.1. The polar nodes chosen need not be neighbors of the source node. They are referred to as *poles*, since they are the extreme points or poles of the packets as the packets propagate towards their destination. Section 4.5 justifies the reason for choosing between 0 and 2 poles.

The packet to be transmitted is encrypted with the public key of the destination, followed by the public keys of each of the poles in the reverse order of selection. Any public key system such as RSA [14] can be used for this purpose. The packet is then transmitted with the destination set to the first polar node (or the ultimate destination, if no polar nodes were chosen).

We impose the condition that no two consecutive poles can be identical (see Section 4.5). Therefore, if a pole, on decrypting a packet, finds the next destination to be its own address, it will know that it is the ultimate destination of the packet, and that the packet contents are unencrypted. For example, consider the sample network shown in Figure 1(a) where a packet is transmitted from source S to destination D through poles M and U. The packet is encrypted three times, first with the public key of D, followed by that of U, and finally with the public key of M. The packet transmitted by S has M as its destination, with the contents of the packet encrypted with public key of

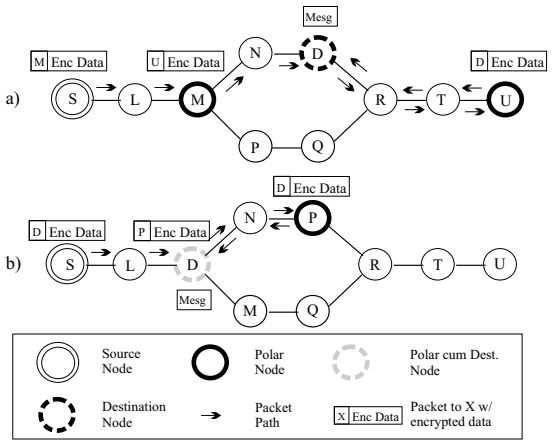

Figure 1: The figure shows two scenarios that encourage (a) Non-Polar and (b) Polar nodes to forward packets destined for other nodes. These scenarios are called *loopback*, since the packet loops back to a node that had previously forwarded it.

M. The packet is sent to M though L. Since the packet is signed with the public key of M, no other intermediate node can interpret the packet contents.

4.3.2 Operation at the Non-Polar Nodes

A non-polar node does not perform any significant function apart from forwarding packets to its neighbor en route to the packet's destination. The possibility of the node being the ultimate destination of the packet provides incentive for the node to forward the packet.

A scenario where a non-polar node forwarding the packet node is the ultimate destination of the packet is depicted in Figure 1(a). In the figure, D is a non-polar node that forwards the packet to U, which is merely a pole. The packet, after passing through U, returns back to D. If D, behaving selfishly, had dropped the packet, it would have lost packets destined for itself.

4.3.3 Operation at the Polar Nodes

with its private key. The decrypted packet contains the address of the next pole/destination and some data. If the address of the following pole is the same as the current node's address, then the data contains the unencrypted message to be delivered to this node; this node is the ultimate destination of the packet. Otherwise, the next pole is either another pole or the ultimate destination of the packet, and the data contains the packet to be transmitted, encrypted with the public key of the next pole. This packet is transmitted to the next pole.

Consider the scenario in Figure 1(b), where a polar node forwarding the packet ends up as the ultimate destination of the packet. After receiving the packet, polar node D decrypts it with its private key and determines

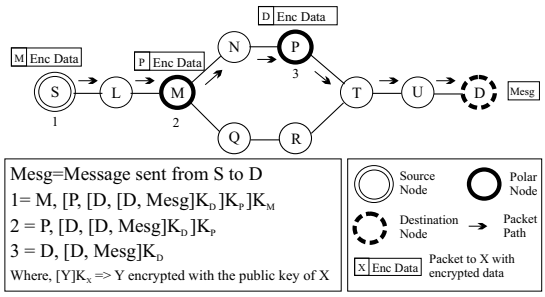

Figure 2: Transmission of $Mesg$ from node S to node D through poles M and P. The rectangles above the node indicate the packet as seen by the node, while the number below the node corresponds to the actual packet contents as indicated in the Legend.

that the packet should be forwarded to P. P, after receiving the packet, decrypts it to find that the packet should be forwarded to D. This decrypted packet contains the packet to be forwarded, encrypted with the public key of D. Finally, after node D decrypts the packet, it discovers that it is also the next hop. It thus knows that it is the destination of the packet. If D, behaving selfishly, had dropped the packet instead of forwarding it to P, it would have lost its own packet.

4.3.4 Operation of AD-MIX

Figure 2, where a packet is transmitted by source S to destination D, with poles M and P. At source S, a dummy header with D as destination is appended to the message $Mesg$ to be transmitted to D. This is then encrypted with the public key of D, followed with the public key of P, and finally with the public key of M. This is represented by '1' in the legend of the figure. The packet transmitted by the source has M as its destination. Non-polar node L, after receiving the packet, forwards it to M. Since the packet is encrypted with the public key of M, L is unable to decrypt its contents. Polar node M, being the intermediate destination of the packet, decrypts it with its public key to find the message to be forwarded to P. The content of this decrypted message is depicted by '2' in the legend. Since this message is encrypted with the public key of P, it cannot be decrypted by M. Polar node M forwards this packet to P, via non-polar node N. Polar node P, after receiving the packet, decrypts it and finds that the packet has to be forwarded to D. The content of the decrypted packet is shown by '3' in the legend. This packet, after being forwarded by non-polar nodes T and U, reaches destination D. Node D, after decrypting the packet, finds the next forwarding address (in the dummy header) to be D again. Hence, it knows that it is the destination of message.

4.4 Other Strategies Employed by AD-MIX

The basic strategy described above is sufficient to encourage participation. In this section, we motivate the need and present strategies to improve the efficiency of AD-MIX. These optimizations result in significant improvements to the performance of the protocol.

4.4.1 DSR as the Routing Protocol

For an ad hoc network with n nodes running AD-MIX, the worst case hop count of a packet is $3 * (n - 1)$. A packet traversing the worst case path will cause a substantial increase in the delivery time if an on-demand routing protocol is used. Also, choosing poles randomly results in a low probability of sources/poles having a route to the next pole/destination. For reactive routing protocols like AODV [13] and DSR [10], this may result in up to two additional route discoveries per packet to determine the route to the next pole, and possibly another route discovery to determine the route from the final pole to the destination.

Thus, the choice of polar nodes significantly affects the performance of the protocol. It is advantageous to choose poles along the route to the destination so that there is no significant increase in path length. Choosing poles along the route will also result in fewer route discoveries to find the route to the next pole/destination in reactive protocols. Hence, it is advantageous to employ source-routing protocols like DSR [10] or AODV-PA [9] that maintain the complete route to destinations.

Dynamic Source Routing (DSR) is a source routing protocol proposed for MANET. It maintains a route cache that contains multiple, complete source routes to destinations. Therefore, instead of choosing polar nodes randomly, the source node constructs the set of all nodes along routes that pass through the destination or terminate at the destination. This set is called the *Polar Node Set*. The polar nodes are then chosen from this set.

Figure 3 shows the path traversed by packets corresponding to the poles chosen. If node S wishes to transmit packets to node D, and at S the routes that pass through or terminate at D are $S \rightarrow L \rightarrow M \rightarrow N \rightarrow P \rightarrow D$ and $S \rightarrow L \rightarrow M \rightarrow Q \rightarrow R \rightarrow D \rightarrow U \rightarrow T$, then the *Polar Node Set* corresponding to D would be $\{L, M, N, P, Q, D, R, U, T\}$. Hence, poles would be chosen among these nodes. Choosing M and P as the poles (Case (a)) will not cause an increase in the path length, while choosing Q and U (Case (b)) as the poles will result in a slight increase in path length. Notice that not choosing poles in the direction of the destination will always result in a slight increase in the path length. For example, in Figure 3(a), choosing pole P

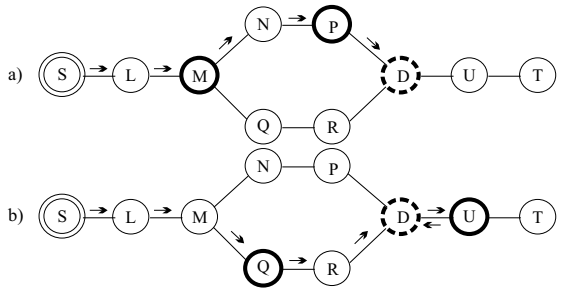

Figure 3: Examples of sets of poles and the corresponding path of packets, where node S is the source and node D the destination. The arrows denote the path of the packets.

followed by M for a packet, would result in a greater path length than choosing M followed by P.

The advantages of employing source-route based routing protocols such as DSR are:

- The number of hops to reach the destination is close to optimum, since poles be chosen along the path to the destination.
- Since the poles are chosen from DSR's route cache, it is very likely that routes to these nodes are already known. Thus, the number of new route discoveries to the poles is minimal.

4.4.2 Caching and Pre-determining Poles

For a source and destination pair, the *Polar Node Set* remains fairly constant over short time intervals. Also, communication between nodes is usually bursty or continuous, rather than sporadic. Therefore, it is advantageous to cache the *Polar Node Set* and update it at regular intervals, rather than reconstruct it from the route cache poles for every packet transmitted.

A further improvement to this strategy is to cache a subset of possible poles for each destination in a *Pre-selected Polar Node Table (PPNT)*, as opposed to caching the entire *Polar Node Set*. The *Polar Node Set* to each destination is constructed at the beginning of each interval. The entries of the PPNT are then populated with 0, 1 or 2 randomly chosen poles picked from the *Polar Node Set*. The PPNT is constructed only once at the beginning of each interval. Hence, instead of choosing the polar nodes from the *Polar Node Set* for each packet, the source node randomly picks an entry from the PPNT that contains the poles to be used for the packet. An upper bound is imposed on the number of entries in the PPNT to limit its size. The number of different routes that can be chosen for a destination is limited by this bound. Table 1 shows a sample PPNT for the network shown in Figure 3.

The main advantage of using the PPNT is that all packets to a destination are sent through only a few

Num. of Poles	Pole 1	Pole 2
1	M	**N/A**
2	M	P
1	T	**N/A**
2	P	M
2	R	P
0	**N/A**	**N/A**

Table 1: Example Pre-selected Polar Node Table (PPNT) for node D at node S.

routes dictated by the entries of the PPNT. This results in fewer route discoveries, and consequently lower control overhead and packet delivery time. Also, less time is spent on deciding the number of poles and choosing polar nodes. Thus, utilizing the PPNT further reduces the packet delivery time. In fact, if only the header of the packet is encrypted as opposed to the entire packet, the encrypted header can be pre-calculated and stored in the PPNT. This will result in significantly lower delay and processing overhead, since the process of encrypting the packet header with the public key of each pole is performed just once at the beginning of each interval, instead of for every packet transmitted.

AD-MIX periodically refreshes the contents of the PPNT to accommodate changes in the route cache due to dynamic changes in network topology. In order to accommodate dynamic changes in network topology, it is beneficial to vary the refresh rate with the mobility of the node and the number of RERRs received.

4.4.3 Forcing loopback

A selfish node is forced to forward packets that are not immediately destined for itself because there is a possibility that it is the ultimate destination of these packets; i.e., these packets can loop back to the node. If the possibility of packets looping back is very small, then a selfish node may be tempted to risk dropping packets not destined for it. To prevent this, AD-MIX can force a specified percentage of the packets to loop back. Increasing the percentage of packets that loop back increases the probability of a selfish node dropping its own packets. This results in increased cooperation in forwarding packets destined for other nodes.

AD-MIX forces loopback by either choosing a node beyond the destination, with the destination node along its path as a pole; or, if no such node is known, choosing the destination as the first pole and any node along the path as the second pole.

For example, Figure 4(a) illustrates how loopback is enforced when a node beyond the destination is known to the source. If node S wants to communicate with D, and it has a path to T containing D along it, then

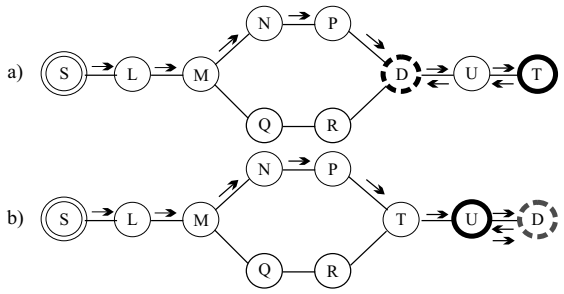

Figure 4: Two cases of forced loopback.

it can force loopback by choosing T as the pole. If D is selfish and drops the packet after noticing that the destination is T, it will drop its own packet. However, if S does not know any node beyond D, then it would not be able to employ the previous scheme. In such a case, the source can send looped back packets with two poles. The first pole is the destination itself, while the second pole could be any node along the path. This is depicted in Figure 4(b), where U is chosen as the second pole. Here again, if the node behaves selfishly, it will drop packets destined for itself.

The variations of AD-MIX evaluated in Section 5 of this paper implement all the optimizations mentioned previously, with 0% (no forced loopback), 10% and 30% forced loopback.

4.5 Encouraging Participation using AD-MIX

In the previous sections we discussed the operation of the AD-MIX protocol. In this section, we show that the number of poles required to encourage participation should be ≥ 2, if a constant number of poles are used for all packets transmitted. Alternatively, any number of poles, between 0 and 2 (inclusive) can be chosen, if a variable number of poles is used for each packet.

To show that AD-MIX encourages participation, it is sufficient to show that it is impossible for any node to deduce the ultimate destination of any packet passing through it. We now prove that a minimum of two poles is necessary to achieve this.

- *Zero poles:*
  If the packets are not encrypted, or encrypted with just the public key of the destination, then a selfish node can determine whether it is the intended destination of the packet by looking at the destination address. It can drop the packet if it is not the destination.
- *One pole:*
  If every packet is encrypted with the public key of one pole, then the non-polar nodes would have no way of determining the true destination of the packet, since the destination specified in the packet header may either be the next pole or the destination. Since it is possible that the non-polar node itself could be the destination, a selfish non-polar node is forced to forward packets not destined for itself, or risk dropping its own packets. However a polar node, after decrypting the packet, can determine that the packet is not destined for it. Since only one polar node is used, the node to which the polar node would forward the decrypted packet has to be the packet's ultimate destination. Hence, a selfish polar node could drop the packet without the risk of losing messages sent to it.
- *Two or more poles:*
  When two or more poles are used, it becomes impossible for even the polar nodes to determine the ultimate destination of the packets they receive. By padding the header with variable number of blank poles, it is possible to conceal the number of times a packet is encrypted, since this information cannot be obtained from either the size or the destination of the packet. Therefore, a polar node cannot be sure, even after decrypting the packet, whether the next address of the packet is the destination, unless it has information to suggest that it is the second pole of the packet. Since no node possesses this information, the destination of packets are successfully obscured.

Hence, in order to conceal the destination of the packet from both polar and non-polar nodes, at least two poles must be chosen.

Nonetheless, using two or more poles for each packet will incur a considerable control overhead and delay in mobile ad hoc networks. AD-MIX overcomes this by choosing variable number of poles (between 0 and 2) for each packet. Hence, even if no poles are chosen or if only one pole is chosen for a particular packet, the intermediate nodes will be forced to propagate the packet since they cannot deduce the number of poles chosen for the packet. A node cannot rule out the possibility of the packet looping back to it.

Choosing two identical poles for a packet

If both the poles chosen for a packet are identical, AD-MIX's assumption that the intermediate nodes do not have sufficient information to deduce the packet's ultimate destination is broken. If a pole decrypts a packet twice, it will know that the node it should forward the packet to is the packet's ultimate destination. A selfish pole can therefore drop such packets. To prevent this, a route should never use two identical poles.

5 Performance Evaluation

In this section, we evaluate the variations of AD-MIX to study the performance of each under a variety of conditions.

5.1 Simulation Model

In order to evaluate the AD-MIX protocol, we use the Network Simulator (NS-2) [8] with wireless extensions. Our simulation parameters are detailed in Table 2.

Parameter	Value
Source Agent	CBR
Routing Layer	DSR
MAC Layer	IEEE 802.11 DCF
Channel Bit Rate	2 Mbps
Queue Type	Priority Droptail
Queue Length	50
Radio Range	250m
Radio propagation model	Two-Ray Ground

Table 2: Simulation Parameters.

We compare the performance of AD-MIX with a protocol that does not have any mechanism to encourage participation. We also evaluate the effectiveness of the variations of AD-MIX. The goals of our simulations include:

- Studying the overhead caused by AD-MIX and determining its impact on the network performance.
- Evaluating variations of AD-MIX to determine the variation with the best overall performance.
- Determining the effect of selfish nodes in the network by comparing the performance of the network with and without AD-MIX in the presence of 10%, 20% and 30% selfish nodes.

The various protocols being compared and their importance are described below.

DSR: This consists of a network with DSR as the routing protocol. This is used as a base case to compare against AD-MIX and its variants, with no mechanism to encourage participation.

AD-MIX (without forced loopback): This variant of AD-MIX does not force loopback on the packets transmitted. Packets may still loopback due to the selection of poles; it is merely not imposed by the protocol.

AD-MIX with 10% forced loopback: This variation of AD-MIX forces at least 10% of the packets transmitted to loopback. Looping back increases the chances that a node forwarding a packet is the ultimate destination of the packet, thus encouraging greater participation among nodes.

AD-MIX with 30% forced loopback: This variation of AD-MIX is similar to the one described above except that at least 30% of the transmitted packets are forced to loopback instead of 10%.

In order to find the optimal PPNT refresh rate, we evaluated the performance of AD-MIX with 30% forced loopback with varying refresh rates. While not shown, simulation results showed that AD-MIX performs best at a refresh rate of 10 seconds for PPNT. Hence, for our evaluation, we chose a fixed refresh rate of 10 seconds for the PPNT, and the upper limit of the number of entries in the PPNT was set to 10. To study the performance of AD-MIX in the presence of selfish nodes, we also tested AD-MIX with 30% forced loopback in a network with 50 nodes with a varying number of selfish nodes.

5.2 Simulation Metrics

The performance of AD-MIX was evaluated based on the following metrics:

Packet Delivery Ratio: The fraction of data packets sent by the source that are received at the destination.

Control Overhead: The number of control packets transmitted during the simulation. This is the overhead needed to discover and maintain paths to other nodes.

Packet Delivery Time: The average time taken for a data packet to reach the destination. This includes the route discovery latency, the packet processing time at the intermediate nodes and encryption and decryption time.

Path Length: The average number of hops each data packet traverses to reach the destination.

5.3 Simulation Topology

In our simulations, we used the random waypoint model with the pause time set to 30 seconds. Performance analysis of the protocols were conducted under identical scenarios with the nodes moving at velocities between 0 m/s and 10 m/s for a network with 50 nodes. The details of the network topology used for the simulations are listed in Table 3.

Simulation results were averaged over 10 runs, each with different initial position and movement of nodes. The duration of the simulations was 500 seconds.

Parameter	Value
Number of nodes	50
Network area (m^2)	1000x1000
Number of data sessions	15
Packet size	512 bytes
Rate of transmission	4 packets/second
Number of packets per session	1300

Table 3: Simulation Parameters.

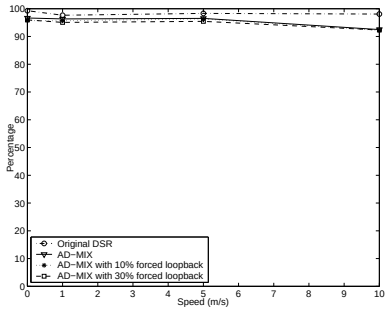

Figure 5: Packet Delivery Ratio.

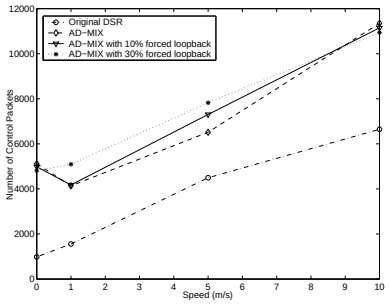

Figure 6: Control Overhead.

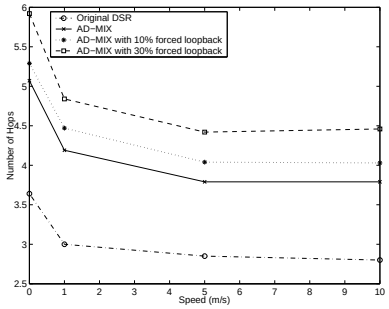

Figure 7: Path Length.

5.4 Simulation Results

5.4.1 Packet Delivery Ratio (PDR)

PDR is the measure of the percentage of data packets sent by the source that were received at the destination. This metric indicates the performance of the protocol.

Figure 5 shows the PDR of the evaluated networks. The PDR of networks using variations of AD-MIX are about 95%, even at speeds of 10 m/s. In comparison, the PDR of network using DSR is around 98%. Hence, employing ADMIX does not cause a significant degradation in the network performance, even under high mobility.

5.4.2 Control Overhead

AD-MIX requires packets to pass through a set of polar nodes before reaching the destination. This often causes additional control message floods to be generated to determine routes to these poles, if a route to them is not already known. Hence AD-MIX should cause an increase in the amount of control packets generated. The control overhead of the various protocols is shown in Figure 6. Predictably, AD-MIX results in higher control overhead than DSR. AD-MIX causes nearly twice as many route discoveries as DSR. The replies generated in response to these requests are considerable, hence there is an increase in the number of control messages generated.

The performance of AD-MIX with 10% and 30% forced loopback is not much worse than AD-MIX without forced loopback, since the poles are selected from the nodes in the route cache. Therefore, there is a high probability that routes to the poles already exist. AD-MIX with forced loopback causes few additional route discoveries in comparison with AD-MIX without forced loopback. Thus, improved participation realized by forced loopback is achieved without any significant further degradation in performance.

5.4.3 Path Length

Looping back packets and not choosing poles in the direction of the destination results in AD-MIX having a greater path length than a network using only DSR. Figure 4 depicts a few instances where employing AD-MIX would cause an increase in path length.

The average number of hops a data packet traverses to reach its destination is shown in Figure 7. Not surprisingly, AD-MIX causes an increase in the average path length. To investigate whether this increase varies with different size networks, we compare the hop count of AD-MIX with DSR in networks of varying size. This is represented in Figure 8. We notice that even for networks as large as 100 nodes, AD-MIX causes an increase of only 2 hops more than that of DSR. This can be attributed to the fact that poles chosen are always along the path to the destination. Also, the effect of forced loopback on the number of hops taken to reach the destination is not very significant.

The unusually high path length for the static configuration is due to the random initial positions of the nodes, which does not change with time.

5.4.4 Packet Delivery Time (PDT)

Packet Delivery Time (PDT) is the average time taken by a data packet to reach the destination. It is a measure of the processing and other overhead caused by the protocol.

NS-2 does not take into account the packet processing time at each intermediate node. In order to determine a more realistic value of PDT, we consider the time required to process a packet at each intermediate node that the packet passes through, and the time spent in encryption and decryption of the packets.

Field tests conducted by our research group on the AODV routing protocol have found the packet processing time at each node to be around 2 ms. We assume that a network using DSR as its routing protocol would incur a similar per-node packet processing time. Further, based on previous results in [15], we

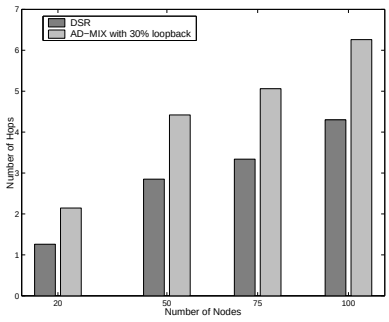

Figure 8: Variation of path length.

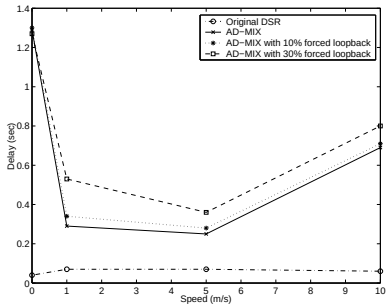

Figure 9: Packet Delivery Time.

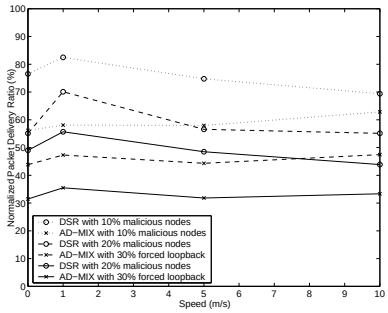

Figure 10: Normalized Packet Delivery Ratio.

assume an average encryption time of 8.5 ms and an average decryption time of 0.5 ms.

Using these numbers, Figure 9 depicts the mean packet delivery time of the protocols, including per-node packet processing time and encryption and decryption times. The figure shows that AD-MIX causes an increase in packet delivery time in comparison to DSR. This can be attributed to the fact that the AD-MIX protocol forces packets to pass through a series of poles before reaching the destination, resulting in an increase in path length, and hence the mean packet delivery time. Also, choosing poles from routes that are no longer valid due to changes in network topology results in new route discoveries. This introduces additional delays, which in turn increase the packet delivery time. Finally, the time required for multiple encryptions of packets at the source node and decryptions at the poles and destination also increases the packet delivery time. Therefore the PDT of AD-MIX is greater than that of DSR.

Again, the reason for the relatively high PDT in the static scenario is due to the high static path length.

5.4.5 Impact of selfish nodes on the network

The following simulations show the impact of selfish nodes in the network. To simulate selfish behavior in nodes, in our first experiment we adopt a model similar to the one used by [12], where selfish nodes agree to participate in packet forwarding by cooperating in forwarding control messages, but later drop data packets destined to all other nodes. While selfish nodes receive packets destined for themselves, their presence adversely affects the functioning of the network, since they disrupt data delivery to other nodes.

We use Normalized Packet Delivery Ratio as the metric to evaluate the performance of AD-MIX with 30% forced loopback in the presence of selfish nodes. We define Normalized Packet Delivery Ratio (NPDR) as the ratio of the PDR of the network in the presence of selfish nodes to the PDR of the network without selfish nodes. Hence, NPDR is the measure of degradation of the network in the presence of selfish nodes. For example, a NPDR of 80% implies that the protocol delivers 80% of its maximum capability in the presence of selfish nodes. Also, a lower value of NPDR implies greater loss in a network containing selfish nodes.

Figure 10 shows the NPDR of *only* the selfish destinations with and without the presence of AD-MIX in a 50-node network with 10%, 20% and 30% selfish nodes. In this experiment, we evaluate the penalty in the delivery ratio to the selfish nodes, assuming those nodes *continue* to be selfish. There is a significant decrease in the NPDR of selfish nodes when AD-MIX is employed, when compared to a network that does not use any mechanism to prevent selfishness. Selfish nodes drop some of their own packets, since they mistake these packets as being destined for other nodes. Consequently, the selfish nodes experience a lower PDR. In fact, if AD-MIX with 30% forced loopback is employed, the maximum NPDR a selfish node can hope to achieve is 70%, since AD-MIX with 30% forced loopback will cause the selfish nodes to drop *at least* 30% of their own packets. In comparison, the performance of the non-selfish nodes in the network will not be as significantly affected since they do not drop their own packets. While not shown, this results in higher throughput to the non-selfish destination nodes.

Figure 11 depicts the performance benefits achieved by using AD-MIX in the presence of selfish nodes in the network. In this experiment, we assume that the AD-MIX protocol *prevents* selfish behavior. Hence, while the PDR of networks not employing AD-MIX decreases drastically with increase in the percentage of selfish nodes in the network, the performance of AD-MIX remains constant, regardless of the percentage of selfish nodes in the network.

We observe from Figure 10 that employing AD-MIX in a network containing nodes that continue to be selfish results in a marked degradation in the

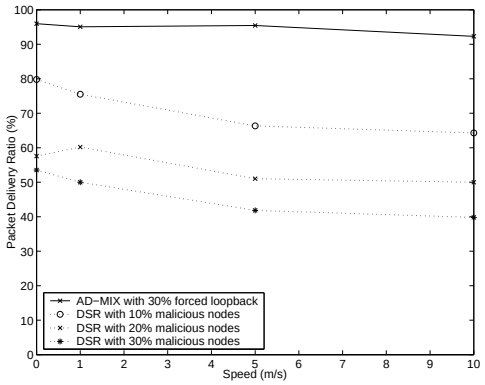

Figure 11: Packet Delivery Ratio of the entire network.

performance of the selfish nodes, while Figure 11 reveals that the incentives provided by AD-MIX result in a significantly higher PDR. The improved performance achieved by benign behavior, when AD-MIX is employed, can serve to mitigate selfish behavior.

5.4.6 Summary of Simulation Results

We infer from the evaluation of our simulation results that AD-MIX protocol performs well, with its packet delivery ratio around 95% even when the node mobility is 10 m/s. Hence, AD-MIX does not cause a significant degradation in performance despite providing incentives for participation of nodes. Also, the performance of AD-MIX does not degrade with increased mobility. AD-MIX incurs additional overhead in terms of packet delivery time and number of hops required to reach the destination. However, the increase in the number of hops over DSR is only about two hops even for networks containing up to 100 nodes. Another significant feature of AD-MIX is that forced loopback causes only a slight increase in overhead in comparison with AD-MIX without forced loopback. This is primarily due to the clever choice of polar nodes while enforcing loopback. We also notice that employing AD-MIX in a network containing selfish nodes results in a marked decrease in the performance of nodes that choose to behave selfishly. Otherwise, if AD-MIX succeeds in preventing selfishness, the network performance remains high.

6 Security and Other Considerations

In this section we discuss attributes of AD-MIX that enable it to achieve more than encouragement of participation. We also discuss circumstances under which AD-MIX may not be able to achieve its objectives effectively.

6.1 Secure Communication

Since the source encrypts the contents of a packet up to three times before transmitting it, at no instant along the route are the contents of the packets available unencrypted. Only the final destination can decrypt the packet to view its contents. The protocol design therefore provides information security as well as prevents *man-in-the-middle* and *replay attacks*. Hence AD-MIX provides a secure path for communication of messages from the source to the destination.

6.2 Desire to Communicate

AD-MIX assumes that all nodes in the network want to participate in network communications. This is a reasonable assumption since the purpose of a node joining an ad hoc network is to communicate with other nodes in the network. However, if a particular node does not expect to receive any packet, then it can drop all packets passing through it. Such nodes may be classified as malicious. We assume that other mechanisms that detect packet drops, such as [2], are utilized to detect and handle such nodes.

6.3 Colluding Nodes

If two nodes collude by agreeing not to use any poles to transmit data between themselves, then the destination node can safely drop all packets not destined to it, since it is sure that the source does not use any poles while transmitting to it. Under such circumstances, the current design of AD-MIX will not work. However, such colluding nodes can be handled in two ways:

- Use other mechanisms, such as [2], to detect nodes that do not forward packets on behalf of other nodes in the network. The action taken against colluding nodes may vary from merely reporting misbehavior to isolating them from the network. The action taken should deter other nodes from colluding.
- Assume the presence of a 'tamper-proof' layer at each source node that cannot be by-passed. All packets transmitted by the node are forced to pass though this layer. This layer decides the set of poles for each transmitted packet. This prevents collusion since the choice of polar nodes is no longer under the control of the user of the source node.

6.4 Discarding Control Packets

A node may decide to drop all control packets passing through it except route request packets destined for itself. If this happens, no node in the network will have any routes *through* this node; any route with the node's address will be a route terminating at the node. This implies that if AD-MIX is not used, then by dropping all control packets through the node, it can

ensure that no packet will ever be sent to it unless the packet is destined for it. A selfish node can thereby save its resources. However, employing AD-MIX mitigates this behavior since if the source node possesses a path to the selfish node, it can use the selfish node as a pole to transmit packets to other nodes along the path.

6.5 Traffic Analysis and Anonymization

Since AD-MIX employs a model similar to the mix servers, packets traveling between a source and a destination do not follow the same path. Even if they follow the same path, choosing different poles will make the packets appear dissimilar, making it extremely difficult to gather information about on-going sessions by traffic analysis. Hence, AD-MIX also provides an effective mechanism to counter attacks against privacy. Also, since the destination of the packet is hidden from all nodes in the network, AD-MIX inherently promotes anonymization.

7 Conclusion

Mobile ad hoc networks, due to their unique architecture and mode of operation, pose networking challenges not encountered in other networks. Selfishness, maliciousness and lack of trust are some of the challenges unique to ad hoc networks. Prevention of selfishness and malicious behavior is particularly important since it can adversely affect the performance of the network.

In this paper, we have addressed the issue of selfishness of nodes in mobile ad hoc networks. We identify the requirements of a protocol to encourage participation. Based on these requirements, we propose a new protocol, called AD-MIX, to stimulate cooperation among nodes by adversely affecting the performance of selfish nodes. Our simulation results show that, despite providing incentives for participation of nodes, the performance of AD-MIX in terms of packet delivery ratio was found to be nearly as good as that of DSR, even under high node mobility. Also, by choosing polar nodes appropriately, overhead caused by forced loopback can be reduced. Thus, improved participation realized by forced loopback is achieved without any significant degradation in performance. Evaluation of AD-MIX in a network with selfish nodes shows that AD-MIX causes a substantial degradation in the performance of selfish nodes, thus forcing selfish nodes to participate. However, AD-MIX introduces additional overheads due to encryption and decryption of packets, and increase in packet size due to multiple headers. This overhead is necessary to ensure fair behavior of nodes.

References

[1] N. Asokan and P. Ginzboorg. Key-Agreement in Ad-hoc Networks. *Computer Communications*, 23(17):1627–1637, November 2000.

[2] S. Buchegger and J. Y. LeBoudec. Nodes Bearing Grudges: Towards Routing Security, Fairness, and Robustness in Mobile Ad Hoc Networks. In *Proceedings of the Tenth Euromicro Workshop on Parallel, Distributed and Network-based Processing*, pages 403 – 410, Canary Islands, Spain, January 2002.

[3] S. Buchegger and J. Y. LeBoudec. Performance Analysis of the CONFIDANT Protocol: Cooperation Of Nodes – Fairness In Dynamic Ad-hoc NeTworks. In *Proceedings of IEEE/ACM Symposium on Mobile Ad Hoc Networking and Computing*, pages 226–236, Lausanne, Switzerland, June 2002. IEEE.

[4] L. Buttyán and J. P. Hubaux. Enforcing Service Availability in Mobile Ad-Hoc WANs. In *Proceedings of the First IEEE/ACM Workshop on Mobile Ad Hoc Networking and Computing (MobiHOC)*, pages 87–96, Boston, MA, USA, 2000.

[5] L. Buttyán and J. P. Hubaux. Nuglets: Virtual Currency to Stimulate Cooperation in Self-Organized Mobile Ad Hoc Networks. Technical Report DSC/2001/001, Swiss Federal Institute of Technology, Lausanne, Switzerland, January 2001.

[6] S. Capkun, L. Buttyan, and J. Hubaux. Self-organized public-key management for mobile ad hoc networks, 2002.

[7] D. L. Chaum. Untraceable Electronic Mails, Return Addressed and Digital Pseudonyms. *Communications of ACM*, 24(2), February 1981.

[8] K. Fall and K. Varadhan. *The NS Manual. The VINT Project.* A Collaboration between researchers at UC Berkeley, LBL, USC/ISI, Xerox PARC, November 2001.

[9] S. Gwalani and E. Belding-Royer. AODV-PA: AODV with Path Accumulation. In *Proceedings of the Next Generation Internet Symposium, held in conjunction with ICC, IEEE*, May 2003.

[10] D. B. Johnson and D. A. Maltz. Dynamic Source Routing in Ad Hoc Wireless Networks. In Imielinski and Korth, editors, *Mobile Computing*, volume 353. Kluwer Academic Publishers, 1996.

[11] A. Khalili, J. Katz, and W. A. Arbaugh. Toward Secure Key Distribution in Truly Ad-Hoc Networks. In *IEEE Workshop on Security and Assurance in Ad hoc Networks*, Orlando, FL, January 2003.

[12] S. Marti, T. J. Giuli, K. Lai, and M. Baker. Mobile Computing and Networking. In *Proceedings of the Mitigating Routing Misbehavior in Mobile Ad Hoc Networks*, pages 255–265, 2000.

[13] C. E. Perkins and E. M. Belding-Royer. Ad-hoc On-Demand Distance Vector Routing. In *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, pages 90–100, February 1999.

[14] R. L. Rivest, A. Shamir, and L. M. Adelman. A method for obtaining Digital Signatures and Public-key Cryptosystems. *Communications of the ACM*, 21(2):120–126, February 1978.

[15] K. Sanzgiri, B. Dahill, B. N. Levine, C. Shields, and E. M. Belding-Royer. A Secure Routing Protocol for Ad Hoc Networks. In *Proceedings of 2002 IEEE International Conference on Network Protocols (ICNP)*, Paris, France, November 2002.