

# Scalable Multicasting on - Demand Source Routing for Mobile Ad-hoc Networks

**M.Prasad,**

Assistant Professor

B.V.C Engg College, Odalarevu

E.Mail: Prasad.maddula@gmail.com

**T.Srinivasa Rao**

Assistant Professor

B.V.C Engg College, Odalarevu

E.Mail: srinu\_tumma@yahoo.co.in

***Abstract**-Increasing popularity and availability of portable wireless devices, which constitute mobile ad-hoc networks calls for scalable ad-hoc routing protocols. On-demand routing protocols adapt well with dynamic topologies of ad-hoc networks because of their lower control overhead and quick response to route breaks caused by mobility or battery failures. But as the size of the network increases, these protocols cease to perform due to large routing overhead generated while repairing route breaks. We propose a Multipath on-demand routing protocol (SMORT), which reduces the routing overhead incurred in route recovery by using alternate paths. Further we compare our protocol with its unipath counterpart to show the improved scalability. We modify the well known routing protocol Multicast Ad-Hoc On demand Distance Vector (AODV) to compute fail-safe multiple paths, which include all the intermediate nodes on the primary path with multiple routes to destination. Exhaustive simulations using glomosim with large networks (2000 nodes), confirms that our protocol improves scalability. Glomosim visualization tool simulates the real time, which compares radio layer, network layer probabilities of successful transmissions and compares the protocols of endair A, network IP, TCP, radio account noise, TELNET, FTP.*

## 1. INTRODUCTION

Routing is one of the most basic networking functions in mobile ad-hoc networks. Hence, an adversary can easily

Paralyze the operation of the network by attacking the routing protocol. This has been realized by many researchers and several “secure” routing protocols have been proposed for ad-hoc networks. Routing has two main functions: route discovery and packet forwarding. There are different types of ad-hoc routing protocols. One can distinguish proactive (e.g., OLSR) and reactive (e.g., AODV and DSR) protocols. Protocols of the latter category are also called on-demand protocols. Another type of classification

distinguishes routing table-based protocols (e.g., AODV) and source routing protocols (e.g., DSR). In this paper, we focus on the route discovery part of on-demand source routing protocols.

At a very informal level, security of a routing protocol means that it can perform its functions even in the presence of an adversary whose objective is to prevent the correct functioning of the protocol. Since we are focusing on the route discovery part on-demand source routing protocols, in our case, attacks are aiming at making honest nodes receive “incorrect” routes as a result of the route discovery procedure.

The main contributions of our work are the following

1. The application of the well-established simulation approach in anew context(ad-hoc routing protocols),
2. The discovery of as yet unknown attacks against previously published ad-hoc routing protocols, and
3. The design of a new on-demand source routing protocol for mobile ad-hoc networks called SMORT (Scalable Multicasting On-demand source Routing) protocol, which is the extension of endair A, which is the extension of Ariadne.

Preliminary results of this work have been presented .in this paper, we have gone for the results to a more powerful adversary that controls multiple adversarial nodes and uses multiple compromised identifiers, and we

allow the simultaneous execution of any number of instances of the route discovery protocol. We also present two new attacks against Ariadne, as well as some extensions to the endair A protocol, which is our SMORT protocol.

2. Operation of the Basic Ariadne Protocol with MACs Ariadne has been proposed as a secure on-demand source routing protocol for ad-hoc networks. Ariadne comes in three different flavors corresponding to three different techniques for data authentication. More specifically, authentication of routing messages in Ariadne can be based on TESLA, on digital signatures, or on MACs (Message Authentication Codes). We discuss Ariadne with MACs.

The initiator of the route discovery generates a route request message and broadcasts it to its neighbors. The route discovery message contains the identifiers of the initiator and the target, a randomly generated request identifier, and a MAC computed over these elements with a key shared by the initiator and the target. This MAC is hashed iteratively by each intermediate node together with its own identifier using a publicly known one-way hash function. The hash values computed in this way are called per-hop hash values. Each intermediate node that receives the request for the first time recomputes the per-hop hash value, appends its identifier to the list of identifiers accumulated in the request, and computes a MAC on the updated request with a key that it shares with the target. Finally, the MAC is appended to a MAC list in the request, and the request is rebroadcast. The purpose of the per-hop hash value is to prevent removal of identifiers from the accumulated route in the route request.

When the target receives the request, it verifies the per-hop hash by recomputing the initiator's MAC and the per-hop hash value of each intermediate node. Then, it verifies the MAC of each intermediate node. If all these verifications are successful, then the target generates a route reply and sends it back to the

initiator via the reverse of the route obtained from the route request. The route reply contains the identifiers of the target and the initiator, the route obtained from the request, and the MAC of the target on all these elements that is computed with a key shared by the target and the initiator. Each intermediate node passes the reply to the next node on the route (toward the initiator) without any modification. When the initiator receives the reply, it verifies the MAC of the target. If the verification is successful, then it accepts the route returned in the reply.

#### A. An attack on Ariadne with MACs

Let us consider the network configuration illustrated in the figure given below.

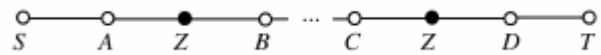


Fig.1. part of a configuration where an active 1-2 attack against Ariadne is possible.

We assume that the adversary controls two adversarial nodes (represented by the black nodes in the figure), and it uses only a single compromised identifier Z. We explain the attack when Ariadne is used with standard MACs, but we emphasize that a similar attack can also be carried out when TESLA is used, or when digital signatures are used and, for efficiency reasons, intermediate nodes do not verify the signature list in the route request.

S initiates a route discovery process toward T. The first adversarial node receives the following route request:

$$\text{msg}_1 = (\text{rreq}, S, T, \text{id}, h_A, (A), (\text{mac}_A)).$$

The adversary does not append the MAC of Z to the request, instead, it puts  $h_A$  on the MAC list, and rebroadcasts the following request:

$$\text{Msg}_2 = (\text{rreq}, S, T, \text{id}, h_A, (A, Z), (\text{mac}_A, h_A)).$$

Recall that the intermediate nodes cannot verify the MACs in the request. Note also that MAC functions based on

cryptographic hash functions output a hash value as the MAC, and therefore,  $h_A$  looks like a MAC. Hence, B will not detect the attack, and the following request arrives to the second adversarial node:

$$\text{Msg}_3 = (\text{rreq}, S, T, \text{id}, H(C, \dots, H(B, h_A)), (A, Z, B, \dots, C), (\text{mac}_A, h_A, \text{mac}_B, \dots, \text{mac}_C)).$$

The adversary removes B... C from the node list and the corresponding MACs from the MAC list. The adversary can do this in the following way: by recognizing identifier Z in the accumulated route, the adversary knows that the request passed through the first adversarial node. By looking at the position of identifier Z in the node list, the adversary will know where,  $h_A$  is on the MAC list. From  $h_A$ , the adversary computes  $h_Z = H(Z, h_A)$  and a MAC on  $(\text{rreq}, S, T, \text{id}, h_Z(A, Z), (\text{mac}_A))$ . And rebroadcasts the following request:

$$\text{Msg}_4 = (\text{rreq}, S, T, \text{id}, h_Z(A, Z), (\text{mac}_A, \text{mac}_Z)).$$

Since the per-hop hash value and both MACs are correct in  $\text{Msg}_4$ , T will receive a correct request, and returns the following reply:

$$\text{Msg}_5 = (\text{rrep}, T, S, (A, Z, D), (\text{mac}_T))$$

When the reply reaches the second adversarial node, it will forward the following message to C:

$$\text{msg}_6 = (\text{rrep}, T, S, (A, Z, B, \dots, C, Z, D), (\text{mac}_T)).$$

Note that B,...,C cannot verify the MAC in  $\text{msg}_6$ . In addition, their identifiers are in the route carried by the reply, and the preceding and following identifiers belong to their neighbors. Therefore, each of them forwards the reply. Finally, when the first adversarial node receives the reply, it removes B... C and one of the Zs from the node list:

$$\text{Msg}_7 = (\text{rrep}, T, S, (A, Z, D), (\text{mac}_T)).$$

In this way, S receives the route reply that T sent. This means that the MAC verifies

correctly and S accepts the route (S, A, Z, D, T) which is nonexistent.

It must be noted that in  $\text{msg}_6$ , the compromised identifier Z appears twice in the node list. Note, however that Ariadne does not specify that intermediate node should check the node list in the reply for repeating identifiers. If each honest node checks only that its own identifier is in the list and that the preceding and following identifiers belong to its neighbors, then the attack works. Moreover, a slightly modified version of the attack would work even if the intermediate nodes checked repeating identifiers in the reply. In that case, the second adversarial node would send the following reply toward S:

$$\text{msg}_6 = (\text{rrep}, T, S, (A, X, B, \dots, C, Z, D), (\text{mac}_T)).$$

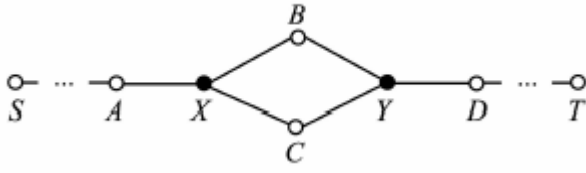
Where X can be any identifier that is different from the other identifiers in the node list. With non eligible probability, X is a neighbor of B, and thus, B will pass the reply on, so that the

First adversarial node can overhear it. Then, the adversary can remove the identifiers X, B... C, and send the reply containing the node list (A, Z, D) to A. A will process the reply, because it contains no repeating identifiers and Z is its neighbor. Alternatively, the first adversarial node may send information about the neighborhood of B to the second adversarial node in a proprietary way.

This is a very powerful attack, because, despite the usage of the per-hop hash mechanism, the adversary manages to shorten an existing route and, therefore, the initiator will probably prefer this short route over others (assuming there are other alternative routes between S and T). In other words, the adversary is able to divert the communication between S and T through itself, and then control it.

#### *B. An attack on the Optimized version of Ariadne*

Let us consider the network configuration illustrated in fig.2. Now, we assume an active 2-2 adversary, meaning that the adversary controls two adversarial nodes (the black nodes in the figure) and uses two compromised identifiers X and Y.



S initiates a route discovery toward T. the first adversarial node receives the following route request:

$$\text{Msg}_1 = (\text{rreq}, S, T, \text{id}, (\dots, A), (\text{mac}_{S\dots A})).$$

The adversary follows the protocol and rebroadcasts the following message:

$$\text{Msg}_2 = (\text{rreq}, S, T, \text{id}, (\dots, A, X), (\text{mac}_{S\dots AX})).$$

Both B and C receive  $\text{Msg}_2$  and rebroadcast the appropriate route request messages, but those are not rebroadcast by the second adversarial node.

Sometime after the first adversarial node broadcast the route request, it creates a fake route reply,

$$\text{Msg}_3 = (\text{rrep}, S, T, \text{id}, (\dots, A, X, B, Y, \dots), (\text{mac}_{S\dots A})).$$

And sends it to B in the name of Y. Since B has processed the route request, it is in a state where it is ready to receive a corresponding route reply. In addition, Y is a neighbor of B, and B is on the node list in  $\text{Msg}_3$ . Therefore, B accepts the reply. Note that  $\text{Msg}_3$  contains the MAC  $\text{mac}_{S\dots A}$ ,

Which was computed by A on the route request, but B does not notice this because intermediate nodes are not supposed to verify MACs in route reply messages (as those are normally computed with a key shared by the initiator and the target of the route discovery).

Next, B forwards  $\text{Msg}_3$  to X. the second adversarial node overhears this transmission, since it is a neighbor of B. In this way, the second adversarial node learns  $\text{mac}_{S\dots A}$ , and now it can generate a route request message,

$$\text{Msg}_4 = (\text{rreq}, S, T, \text{id}, (\dots, A, X, Y), (\text{mac}_{S\dots AXY})),$$

By first computing the MAC  $\text{mac}_{S\dots AX}$  on  $(\text{rreq}, S, T, \text{id}, (\dots, A, X), (\text{mac}_{S\dots A}))$ , With the compromised key of X, and then computing the MAC  $\text{mac}_{S\dots AXY}$  on  $(\text{rreq}, S, T, \text{id}, (\dots, A, X, Y), (\text{mac}_{S\dots AX}))$ ,

With the compromised key of Y. this request is broadcast by the second adversarial node, and it is processed by D and all subsequent nodes.

Since the iterated MAC verifies correctly at the target T, it creates a route reply:

$$\text{Msg}_5 = (\text{rrep}, S, T, \text{id}, (\dots, A, X, Y, D, \dots), (\text{mac}_T)),$$

Where  $\text{mac}_T$  is a MAC computed on the reply with the key shared by S and T. when this reply reaches the second adversarial node, it modifies it as follows:

$$\text{Msg}_6 = (\text{rrep}, S, T, \text{id}, (\dots, A, X, C, Y, D, \dots), (\text{mac}_T)),$$

And sends it to C. since C cannot verify the MAC in the reply, it does not notice the modification made by the second adversarial node. In addition, C has not received any reply yet and, therefore, it accepts  $\text{Msg}_6$  and forwards it to X. then, the first adversarial node removes C from the node list and sends the original  $\text{Msg}_5$  to A. at the end, S receives the same reply sent by T. therefore the MAC verifies correctly and S accepts the route  $(S, \dots, A, X, Y, D, \dots, T)$ , Which is nonexistent.

### C. The basic endair A protocol

T The operation and the messages of endair A are illustrated in fig. in endair A, the initiator of the route discovery process generates a route request, which contains the identifiers of the initiator and the target and a randomly generated request identifier.

$S \rightarrow *$	: (rreq, S, T, id, ())
$A \rightarrow *$	: (rreq, S, T, id, (A))
$B \rightarrow *$	: (rreq, S, T, id, (A, B))
$T \rightarrow B$	: (rrep, S, T, (A, B), (sig <sub>T</sub> ))
$B \rightarrow A$	: (rrep, S, T, (A, B), (sig <sub>T</sub> , sig <sub>B</sub> ))
$A \rightarrow S$	: (rrep, S, T, (A, B), (sig <sub>T</sub> , sig <sub>B</sub> , sig <sub>A</sub> ))

Fig. an example of the operation and messages of endair A. The initiator of the route discovery is S, the target T, and the intermediate nodes are A and B, id is a randomly generated request identifier. Sig<sub>A</sub>, Sig<sub>B</sub>, Sig<sub>T</sub> are digital signatures of a, b, and T, respectively. Each signature is computed over the message fields that precede the signature.

Each intermediate node that receives the request for the first time appends its identifier to the route accumulated so far in the request and rebroadcasts the request. When the request arrives to the target, it generates a route reply. The route reply contains the identifiers of the initiator and the target, the accumulated route obtained from the request, and a digital signature of the target on these elements. The reply is sent back to the initiator on the reverse of route found in the request. Each intermediate node that receives the reply verifies that its identifiers in the node list carried by the reply and that the preceding identifier (or that of the initiator, if there is no preceding identifier in the node list) and the following identifier (or that of the target, if there is no following identifier in the node list) belong to neighboring nodes. Each intermediate node also verifies that the digital signatures in the reply are valid and that they correspond to the following identifiers in the node list and to the target. If these verifications fail, then the reply is dropped. Otherwise it is signed by the intermediate node, and passed to the next node on the route (toward the initiator). When the initiator receives the route reply, it verifies if the first identifier in the route carried by the reply belongs to a neighbor. If so, then it verifies all the signatures in the reply. If all

these verifications are successful, then the initiator accepts the route.

#### 4.SMORT (Practical Extensions to the Basic endair A protocol)

Abbreviated as Scalable Multicasting On-demand Routing for Mobile Ad-Hoc networks. In our model, we made the assumption that the nodes are static. The proof of security of endair A relies on this assumption. More precisely, in the proof, we show that if a route is returned by endair A to an honest node, then that route must exist in the graph that represents the network with overwhelming probability. Moreover, once a route has been returned, it remains valid forever, because the graph does not change. This means that under the assumption of static nodes, the basic endair A protocol is not vulnerable to replay attacks. However, if we relax this assumption, and we allow the nodes to move, then the basic protocol has a problem. In that case when a node initiates a route discovery process and the adversary receives a route request, it can replay an old route reply, and if that reply reaches the initiator, then it will be accepted, despite the fact that it may contain outdated information.

Fortunately, we can easily extend the basic endair A protocol to mitigate this problem. All we need to do is to require the target of the route discovery to insert the random request identifier id in the route reply. Hence, in the extended endair A protocol, the route reply that is passed from intermediate node  $F_i$  to node  $F_{i-1}$  looks as follows:

(rrep,S,T,id,(F<sub>1</sub>,., F<sub>n</sub>),(sig<sub>T</sub>, sig<sub>F<sub>n</sub></sub>,.,.,., sig<sub>F<sub>i</sub></sub>)).

Now, when the initiator receives a route reply, it also verifies if it received back the request identifier that it sent in the route request. This makes it practically impossible for the adversary to successfully replay an old route reply that belongs to a previous route discovery process. of course, when nodes are allowed to move, it is possible that a route reply contains a nonexistent route even if there was no attack at all. In order to alleviate this problem, the time interval within which the

initiator accepts a reply with a specific request identifier should be appropriately limited.

Another problem with the basic endair A protocol is that it is vulnerable to malicious route request flooding attacks. This is because the route request messages are not authenticated in any way and, hence; an adversary can initiate route discovery processes in the name of honest nodes. These forged route discovery processes will be carried out completely, including the flooding of the route requests in the whole network, because only the impersonated initiators can detect that they are forged. In order to prevent this, the route request can be digitally signed by the initiator, and the rate limiting techniques similar to the end user for ariadne can be applied with endair A too. Naturally, such extensions put more burdens on the nodes, since now they also need to verify the initiator's signature in each route request message and to maintain information that is required by the rate limiting mechanism.

Increasing popularity and availability of portable wireless devices, which constitute mobile ad-hoc networks calls for scalable ad-hoc routing protocols. On-demand routing protocols adapt well with dynamic topologies of ad-hoc networks because of their lower control overhead and quick response to route breaks caused by mobility or battery failures. But as the size of the network increases, these protocols cease to perform due to large routing overhead generated while repairing route breaks.

We propose a Multipath on-demand routing protocol (SMORT), which reduces the routing overhead incurred in route recovery by using alternate paths. Further we compare our protocol with its unipath counterpart to show the improved scalability. We modify the well known routing protocol Multicast Ad-Hoc On demand Distance Vector (AODV) to compute fail-safe multiple paths, which include all the intermediate nodes on the primary path with multiple routes to destination. Exhaustive simulations using glomosim with large networks (2000 nodes), confirms that our protocol improves

scalability. Glomosim visualization tool simulates the real time, which compares radio layer, network layer probabilities of successful transmissions and compares the protocols of endair A, network IP, TCP, radio account noise, TELNET, FTP.

GloMoSim (Global Mobile Information System Simulation) is a network simulation package used for wireless simulation. It was designed in order to simulate various large scale wireless networks with fixed or variable mobility under a range of conditions. GloMoSim was developed by the Parallel Computing Laboratory in UCLA by the use of a C like platform PARSEC which is used for sequential and parallel execution of discrete event simulation models. GloMoSim was designed by the use of a layered approach based on the OSI reference model. For each of these layers models can be added easily as long as the messages between each of the layers are the same

GloMoSim uses a text file based interface, in which all options are present and the user chooses certain set up simulation parameters and comments out by the use of the # key parameters which are not applicable to the simulation. This set up file is in the */bin* directory and is called *config.in..* For the application layer a further file is needed *app.conf* which is used to setup the various applications for instance FTP, TELNET etc. In this file the application of choice is added and the various parameters for this application also have to be added. Also several applications can be added for a simulation.

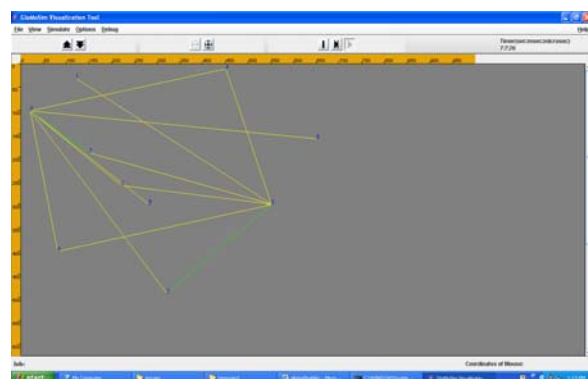


Fig: Glomosim visualization in radio layer

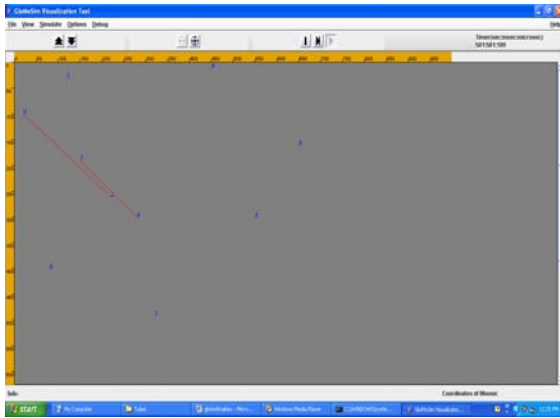


Fig: Glomosim visualization in network layer

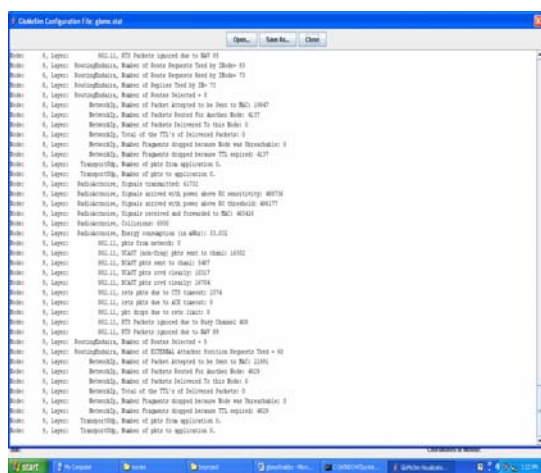


Fig: Glomosim statistics

## Conclusion and Future work

Our paper proposes glomosim simulation result by comparing these protocols at different number of nodes. This comparison result helps in selection of routing protocol for mobile Ad-hoc networks. We ran the simulations in the scenario of 50 nodes, 100 nodes and 200 nodes at different pause times. The scenarios used in the simulation experiments take into account a variety of environmental factors that influence protocol performance. The performance of the protocols is compared in terms of their packet delivery ratio, end to end delay and routing overhead.

From the protocol Ariadne, in which route requests are considered by calculating the per-hop hash values and from endair a always route replays are considered. Any way nodes having mobility in both the protocols such that

source recovery is always difficult. Hence we consider our protocol SMORT (Scalable Multicasting On-demand Routing), in which the nodes are static in order to recover the source. When the nodes are static and whenever route breaks or battery failure occurs, that network edge should be eliminated and there by reducing the number of routes. In order to avoid this once again we can increase the scalability of nodes. Increase in scalability of nodes was supported by this protocol SMORT, since the previous nodes and newly added nodes were kept as static. Our communication model does not abstract away the multi hop operation of the network. In addition, we model the broadcast nature of radio communications, which allows a node to overhear the transmission of a message that was not intended. We also take into account that a radio transmission can usually be received only in a limited range around the sender.

## References

- [1]. Analysis of IEEE 802.11e and application of game models for support of Quality of service in coexisting wireless networks. *By Stefan Mangold. 2003.*
- [2]. Introduction to ANSI/IEEE Std 802.11 1999 Edition.
- [3]. Simulation of IEEE 802.11e in GloMoSim. *By Daragh Kelleher August 2003.*
- [4]. <http://pcl.cs.ucla.edu/projects/parsec/manual/>
- [5]. <http://pcl.cs.ucla.edu/projects/glomosim/GloMoSimManual.html>
- [6]. <http://www.itr.unisa.edu.au/~sgordon/doc/hybridinterwork2002.pdf>
- [7]. <http://pcl.cs.ucla.edu/projects/parsec/>
- [8]. <http://pcl.cs.ucla.edu/slides/workshop99/Mineopw99/sld002.htm>
- [9]. <http://pcl.cs.ucla.edu/slides/workshop99/Jayutpw99/sld001.htm>
- [10]. <http://pcl.cs.ucla.edu/slides/Parsec99.html>
- [11]. <http://pcl.cs.ucla.edu/slides/Parsec99.html>
- [12]. <http://www.sssmag.com/pdf/80211p.pdf>
- [13]. <http://standards.ieee.org/getieee802/>
- [14]. <http://www.cis.ohiostate.edu/~jain/talks/>
- [15]. <http://www.ece.uvic.ca/~fayez/talks/>