

## A Scheme to Control Flooding of Fake Route Requests in Ad-hoc Networks

Jayesh Kataria

Mumbai University, India, jayeshkataria@gmail.com

P.S. Dhekne

BARC, Mumbai, India, dhekne@barc.gov.in

Sugata Sanyal

TIFR, Mumbai, India, sanyal@tifr.res.in

**Abstract-** The use of Mobile Ad-hoc Networks (MANETs) has increased manifold in recent times. Reactive routing protocols like AODV [6] and DSR [7], used in MANETs, flood the network with route requests whenever a new route is to be discovered. This technique of flooding can be easily misused by malicious nodes to disrupt the network. Generally all nodes have a limit beyond which requests cannot be sent. Malicious nodes can easily bypass this limit and send out large numbers of fabricated route requests in the network, flooding other nodes which ultimately waste all of their processing and battery power in forwarding them. As a result, genuine route requests get ignored and many routes do not get a chance to form.

In this paper, we propose a method by which this malicious flooding of route requests can be effectively controlled. We show by means of reasoning and simulation that our scheme enhances the efficiency and throughput of the network.

**Keywords:** Ad-hoc network, wireless, Routing Protocol, Flood Control, AODV

### I. INTRODUCTION

The use of Mobile Ad-hoc Networks (MANETs) has increased manifold in recent times. Security, however, poses a major concern in such networks. Conventional security schemes, used in wired networks, cannot be directly applied here. The lack of a central decision making authority only adds to the problems, as communication in MANETs assumes mutual trust among neighbors. A broad overview of the security issues involved in Ad-hoc networks has been covered in [1].

Many cryptographic approaches like S-AODV [2], Ariadne [3] and SEAD [4] have been proposed to enhance the security of Ad-hoc networks. However, the problem with cryptographic approaches is the increased consumption of limited battery and processing power. Hence, non-

cryptographic approaches tend to provide more efficient and simpler solutions.

In this paper we deal with the problem of malicious flooding of route requests in Ad-hoc networks and provide a non-cryptographic solution for the same. We show how flooding of route requests causes significant throughput degradation and disruption of route formation along with wastage of battery power. We extend the simple scheme proposed in [5] by making it more intelligent and adaptive to the surroundings. For the purpose of our study, we consider the AODV routing protocol [6] and then generalize it to most of the reactive routing protocols.

In Section II we explain the problem that our solution targets. Section III briefly explains the simple scheme in [5] while in IV we explain our proposed scheme. Simulation experiments, proving our solution are presented in Section V. In VI we prove the generality of our results using logical reasoning. Section VII narrates the future work possible and concluding remarks.

### II. PROBLEMS DUE TO ROUTE REQUEST FLOODING

Nodes in an Ad-hoc network using the AODV protocol collect route information using control packets like RREQ (Route Request) and RREP (Route Reply). Whenever a node (source) needs to send data to another node (recipient) to which it does not have a route, it broadcasts RREQ packets containing information about the recipient. These packets are forwarded by other nodes until a valid route is found or timeout occurs.

In order to control the number of RREQs generated by a node, AODV specifies the RREQ\_RATELIMIT, a parameter that defines the maximum number of RREQs a node can generate in one second. However, a malicious node can

choose to ignore this limit and flood the network with a large number of fabricated RREQs which then get forwarded by the neighboring nodes. As these fake RREQs in the network increase, the non-malicious nodes use up their limited RREQ\_RATELIMIT in forwarding those while genuine RREQs are dropped.

The phenomenon explained above has various effects, ranging from inefficient routing to complete blocking of route information. The situation worsens if many such malicious nodes exist in the network. The route forming process is disrupted severely in the vicinity of the malicious node(s) and improves slowly as we move further away from the epicenter. This is because the neighboring nodes keep forwarding the fake RREQs till their RREQ\_RATELIMIT gets exhausted and drop the rest. This problem has also been identified in [1].

We can summarize the effects of malicious flooding as follows:

- Wastage of limited memory resources while maintaining routing table entries for routes that will never be used.
- Wastage of battery and processing power while forwarding the fake RREQs.
- Denial of service to genuine nodes when routes are not formed.
- Creation of longer routes where shorter ones could have been possible. Hence reduced throughput due to increased hop count.

The solution we propose provides a simple and efficient way to curb this malicious activity and its harmful effects.

### III. INITIAL NAÏVE SOLUTION

In this section we summarize the approach proposed in [5]. We extend the same in section IV.

[5] detects and isolates malicious nodes on the basis of the following 3 constants:

- *RREQ\_ACCEPT\_LIMIT (RAL)*: It is the number of RREQs that a node can accept and process from each of its neighbors per unit time. Its purpose is to try and ensure fairness by accepting some RREQs from all neighbors rather than many from just one.
- *RREQ\_BLACKLIST\_LIMIT (RBL)*: It is the threshold value that determines if a particular neighbor is malicious or not. If the no. of RREQs sent by a neighbor per unit time exceeds this value, the neighbor is assumed to be acting malicious and is blocked by the node.
- *BLACKLIST\_TIMEOUT (BT)*: It is the time for which a malicious neighbor gets blocked.

It increases exponentially as the malicious node gets blocked more number of times.

Once a (malicious) node exceeds the *RBL* of its neighbor, it is blocked by that neighbor for *BT*. In effect, this process of blacklisting is performed by all the nodes surrounding the malicious node, thus isolating it. Due to this, fake RREQs are no longer forwarded along further hops and cannot flood the network. Since flooding is prevented, other nodes can process and forward genuine RREQs leading to the successful formation of valid routes.

The drawback in this method, however, is the simplistic assumption of the constants namely *RAL* and *RBL*. These are predefined and cannot be dynamically varied, based on parameters and conditions like the network topology, memory and battery power. Consequently these constants do not always suit all possible scenarios. This can lead to problems like blacklisting of normal nodes or dropping more RREQs than necessary, thus damaging the prospect of genuine route formation while trying to isolate malicious nodes.

### IV. OUR APPROACH

The solution we propose overcomes the drawbacks in the naïve method while successfully isolating malicious nodes.

The crux of the solution lies in the fact that the responsibility of containing the RREQ flood is shifted to the neighbors of the malicious nodes. Since malicious nodes disable RREQ\_RATELIMIT, the (non malicious) neighbors need to do the necessary regulation of the RREQ packets. This regulation prevents fake RREQs from crossing and congesting further hops.

In order to control the flood of fake RREQs and ensure fairness to genuine RREQs, it becomes necessary for each node to divide its RREQ\_RATELIMIT fairly among all its neighbors. This bandwidth that each node allots to its neighbors is updated at regular intervals depending on the number of *Active Neighbors* at that time. An *Active Neighbor* is one that has forwarded at least 1 RREQ packet in the previous 2 intervals. This ensures that bandwidth is not wasted by allotting it to neighbors that may never send any RREQ in that given interval.

If *R* is the RREQ\_RATELIMIT capacity of a node that has *N* active neighbors in the current interval, then we can define the following parameter for any *Active Neighbor* *i*:

$$\bullet \text{ } avg_i = (k * R)/N \quad (1)$$

$avg_i$  is used to regulate the flow of incoming RREQ packets in a given interval. If any neighbor exceeds  $avg_i$  in an interval, all the remaining RREQs sent by that neighbor in that interval are simply dropped. ' $k$ ' is used in Eq.(1) as a scaling

factor to reduce the probability of dropping non-malicious RREQs when N is high. It is based on the simple assumption that not every neighbor will use the entire bandwidth allotted to it during a given interval. Thus 'k' allows an overlapping of the available bandwidth.

As mentioned above, it is possible that the resources allotted by a node to its neighbors may remain unutilized. Therefore, to avoid wastage of bandwidth, occasional bursts are allowed such that almost the entire available bandwidth can be used. However if a neighbor goes beyond this burst rate it is deemed as malicious and blacklisted. Blacklisting a neighbor leads to its RREQ packets getting ignored for *BT* time, mentioned in Section III.

This burst limit denoted by *peak* is defined as:

$$\bullet \text{ peak} = \alpha * R \quad \text{where } \alpha < 1 \quad (2)$$

$\alpha$  is the maximum tolerable utilization of R by a single neighbor.

Eqns. (1) and (2) are used together to regulate the flow of RREQ packets and detect and isolate malicious nodes. We can summarize our approach in the following algorithm.

---

```

Begin
if RREQ is received
    Increment rreq_cnt for that neighbor
    Calculate the peak and avg values at that instant
    if rreq_cnt exceeds value of peak
        Blacklist neighbor and Return
    if rreq_cnt exceeds value of avg
        Ignore all RREQs till current interval ends
Return
End
    
```

---

## V. SIMULATION RESULTS

### A. Simulation Setup

Table 1 contains the simulation parameters.

**Table 1: Simulation Parameters**

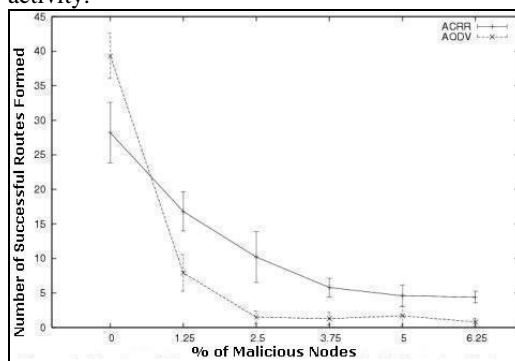
Sr.#	Parameter Type	Value
1	Topology	5000m <sup>2</sup>
2	No. of Mobile Nodes	450
3	Radio Transmission Range	250 m
4	Simulation Time	50 secs
5	Communication Type	FTP
6	Data Packet Size	1000 bytes
7	Confidence Interval	95%

The large rectangular topology enables the formation of long distance routes so that their efficiency can be compared between AODV and our modified protocol which we refer to as *ACRR* (*AODV with Controlled Route Requests*). Parameters 2 and 3 imply a moderate density in the network with approximately 8 to 16 nodes in the interference range. We use NS2 simulator [8]

developed by the VINT project with the CMU wireless extensions [9]. The AODV protocol developed by Uppsala University [10] which conforms to RFC 3561 has been used as the base protocol. Random waypoint mobility is assumed in the network. All the results are averaged over 10 simulation runs. The initial placement of nodes each time was random.

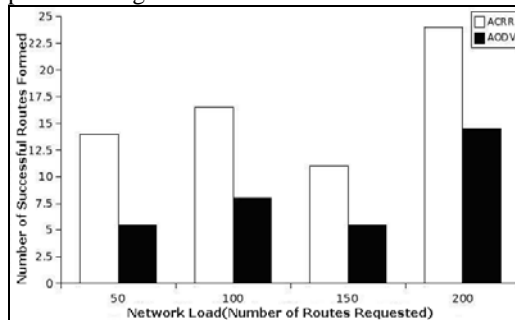
### B. Successful Route Formation

This section compares the number of routes formed in AODV and ACRR with respect to 3 parameters – Malicious Node Count, Network Load and Mobility. The number of routes formed gives a good indication of the effectiveness of the 2 routing protocols under the impact of malicious activity.



**Fig 1: No. of Successful Routes Formed v/s % of Malicious Nodes**

Fig.(1) compares the number of routes formed in AODV and ACRR when malicious nodes are increased from 0% to 6.25% of the total number of nodes. 50 routes are requested during the simulation and the number of routes formed is their subset. Fig.(1) clearly shows that as malicious activity is increased, ACRR is more resistant to the RREQ flooding and forms more routes than AODV since the propagation of genuine RREQ packets is higher in ACRR than in AODV.



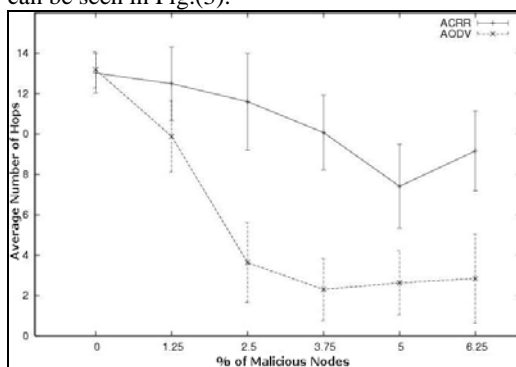
**Fig 2: Number of Successful Routes Formed v/s Network Load. Mobility: 10m/s, 2.5% Mal Nodes**

In Fig.(2) the network load is varied and the response of AODV and ACRR is compared. As the number of route requests generated in the network is increased, the nodes will saturate faster

and drop more packets. In Fig.(2) we can see that the number of routes formed in ACRR is consistently higher than that in AODV and thus proving the scalability of our proposed solution.

### C. Average Route Length

Here, comparison is done between the average length of the routes formed in AODV and ACRR as malicious activity is increased. In both, the protocols requests are made to form routes between far away nodes. Since in AODV genuine route requests are lost in the midst of flooding, these long routes generally timeout before they can be successfully created. Hence in AODV longer routes fail. This problem is tackled in ACRR and can be seen in Fig.(3).



**Fig 3: No. of Hops v/s % of Mal Nodes**

Fig.(3) clearly shows that the average number of hops is higher in ACRR than in AODV. This implies that longer routes stand a higher chance of being successfully formed in ACRR than in AODV.

## VI. EXTENSIONS

The modifications proposed in the earlier sections concentrated mainly on the terminology and messages of the AODV protocol. But the scheme is general enough to work with any reactive routing protocol, as the basic route request mechanism remains the same.

In the simulations, RREQ\_RATELIMIT was assumed to be equal for all the nodes in the network implying that all nodes have a uniform capacity. But heterogeneity can also be easily integrated into the proposed scheme.

## VII. CONCLUSION

We have shown that controlling the flood of route requests in the network using a distributed approach helps in improving the overall performance of the network. The RREQ flow control achieved by using Eqs.(1) and (2) is much better and flexible than the control achieved when using the fixed limits of [5]. Our distributed

approach does not rely on malicious node information dissemination and joint decision making, which makes the scheme well suited for Ad-hoc networks.

## ACKNOWLEDGEMENT

We would like to take this opportunity to thank Punit Rathod without whom this paper would have been impossible. His innovative ideas and insightful suggestions and comments right from the inception of our idea till its implementation have been a major contribution toward the successful completion of this paper.

## REFERENCES

- [1] P. Ning, K. Sun, "How to Misuse AODV: A Case Study of Insider Attacks against Mobile Ad hoc Routing Protocols", *Proceedings of the 4th Annual IEEE Information Assurance Workshop*, 60 (2003).
- [2] M. Zapata, "Secure Ad hoc On-Demand Distance Vector (SAODV) routing", *Internet Draft: draft-guerrero-manet-saodv-00.txt* (2001).
- [3] Y. Hu, A. Perrig, D. Johnson, "Ariadne: A secure on-demand routing protocol for ad hoc networks", *Proceedings of the Eighth Annual International Conference on Mobile Computing and Networking (MobiCom '02)*, 12 (2002).
- [4] Y. Hu, D. Johnson, A. Perrig, "SEAD: Secure efficient distance vector routing for mobile wireless ad hoc networks", *Proceedings of the 4th IEEE Workshop on Mobile Computing Systems and Applications (WMCSA 2002)*, 3 (2002).
- [5] P. Rathod et. al, "Security Scheme for Malicious Node Detection in Mobile Ad Hoc Networks", *6th International Workshop on Distributed Computing (IWDC 2004)*, LNCS, Vol. 3326, 541 (2004).
- [6] C. Perkins, E. Royer, "Ad-hoc On-Demand Distance Vector Routing", *Proceedings of the IEEE Workshop on Mobile Computing Systems and Applications*, 90 (1999).
- [7] D. Johnson, D. Maltz, "Dynamic Source Routing in Ad-Hoc Wireless Networks", *T. Imielinski and H. Korth, editors, Mobile Computing*, 153 (1996).
- [8] The network simulator—ns-2. Available from <http://www.isi.edu/nsnam/ns/>
- [9] Monarch Project, "Wireless and Mobility Extensions to ns-2", <http://www.monarch.cs.rice.edu/cmu-ns.html> (2003).
- [10] Uppsala University implementation of AODV as AODV-UU, <http://core.it.uu.se/AdHoc/AodvUUImpl>