Analysis of Selfish Node Attack in AODV Routing Protocol using GLOMOSIM

[1]Sricheta Ruj, [2]Rishabh Sachdeva
[1]Computer Engineering Department
[1]Nirma University, Ahmedabad, India
[2]Computer Engineering Department
[2] Shri Govindram Seksaria Institute of Technology and Science, Indore, India

***Abstract*—Ad Hoc Networks are extremely vulnerable to attacks due to their dynamically changing topology, the absence of conventional security infrastructures, and vulnerability of nodes, channels and open medium of communication. Selfish nodes in MANETs are the defective nodes which drop the packets that are not intended to them. A malicious selfish node is introduced in the network to analyse the selfish node attack and a trust based algorithm for selfish node attack is also suggested. For the analysis, the routing protocol used in this paper is AODV. Network parameters meters like PDR and end to end delay are evaluated and compared using simulation tool – Glomosim.**

Index Terms— **Mobile Adhoc Network, AODV, selfish attack, glomosim.**

I. INTRODUCTION

An Ad Hoc Network is a collection of wireless computers (nodes), communicating among themselves over possibly multihop paths, without the help of any infrastructure such as base stations or access points [1]. It is an infrastructure-less network [2]. In this, individual network nodes forward packets to and from each other. Due to node mobility, network topology changes frequently So it is important to manage routing information efficiently. Routing protocols can be classified into two types [3] –

Proactive (table driven): This type of protocols maintains fresh lists of destinations and their routes by periodically distributing routing tables throughout the network. The main disadvantages of such algorithms are:

- Respective amount of data for maintenance
- Slow reaction on restructuring and failures

Example, HSR, WRP.

Reactive (on demand): These protocols find a route on demand by continuously sending Route request packets RREQ to the nodes in the network. These have high latency time in route finding. Example, AODV, DSR.

II. AD-HOC ON-DEMAND DISTANCE VECTOR ROUTING PROTOCOL (AODV)

The Ad-hoc On-demand Distance Vector (AODV) routing protocol is a routing proto-col used for dynamic wireless networks where nodes can enter and leave the network. The Ad hoc On Demand Distance Vector (AODV) routing algorithm is a routing protocol designed for dynamic wireless networks. As the name suggest AODV builds routes between nodes as per the wish of source code. AODV is capable of both unicast and multicast routing. The source node transmits a Route Request (RREQ) to its immediate neighbors to find route to a particular destination node. The neighbor replies back with Route Reply (RREP) if the neighbor has a route to the destination. Otherwise the neighbors in turn rebroadcast the request. This continues until the RREQ hits the final destination or a node with a route to the destination. At that point a chain of RREP messages is sent back and the original source node finally has a route to the destination. Here routes are established on demand and the latest route to the destination is found based on sequence number. So the connection setup delay is lower. However, Multiple Route Reply packets in response to a single Route Request packet can lead to heavy control overhead. Here, if source code sequence number is very old, the intermediate nodes may follow inconsistent route and the intermediate nodes have higher but not the latest sequence number leads to stale entries.

Vulnerability of Adhoc Networks.

Nodes of mobile ad hoc networks have limited ranges and because of that it requires multi hop communication. Ad hoc network runs on an assumption that once the node has promised to transmit the packet, it will not cheat but this does not hold true when nodes in the networks have contradictory goals. Node mobility leads to frequent change in network topology. Also, there is Risk of Denial of Service (DoS) attacks due to lack of infrastructure and chances of link breakage and channel errors due to mobility. Nodes in Ad Hoc Networks have limited services and security provision due to limited memory and computational power.

Types of Attack

Black Hole Attack: In mobile ad hoc networks (MANETs), nodes usually cooperate and forward each other's packets in order to enable out of range communication. However, in hostile environments, some nodes may deny to do so, either for saving their own

resources or for intentionally disrupting regular communications [4]. This type of misbehavior is generally referred to as packet dropping attack or black hole attack.

Selfishness Attack: Selfish and malicious nodes participate in route discovery stage properly to update their routing table, but as soon as data forwarding stage begins, they discard data packets.

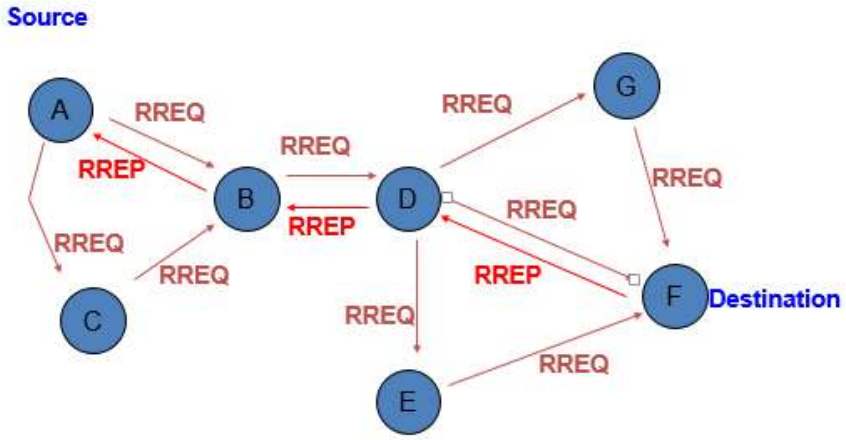

Fig. 1 Message routing of AODV [5]

III. ALGORITHM OF AODV

AODV is a packet routing protocol designed for use in mobile adhoc networks (MANET). It is intended for networks that may contain thousands of nodes. The route discovery mechanism is invoked only if a route to a destination is not known Each node maintains a routing table that contains information about reaching destination nodes. Each entry is keyed to a destination node. Managing the sequence number is the key to efficient routing and route maintenance Sequence numbers are used to indicate the relative freshness of routing information. It is updated by an originating node, e.g., at initiation of route discovery or a route reply. Observed by other nodes to determine freshness. AODV has three different messages that it uses for route discovery and route maintenance. All are sent using UDP.

In this system access AODV Protocol. Some changes required on this protocol. AODV have four different messages that it uses for route discovery and route maintenance. Each AODV router is essentially a state machine that processes incoming requests from the network entity. When the network entity needs to send a message to another node, it calls upon AODV to determine the next-hop. Whenever an AODV router receives a request to send a message, it checks its routing table to see if a route exists. Each routing table entry consists of the following fields [6].

- Destination address
- Next hop address
- Destination sequence number
- Hop count

If a route exists, the router simply forwards the message to the next hop. Otherwise, it saves the message in a message queue, and then it initiates a route request to determine a route. The above Figure 1, flow illustrates this process. Upon receipt of the routing information, it updates its routing table and sends the queued message(s). AODV nodes use four types of messages to communicate among each other.

Sending RREQ

RREQ will only be sent by the source nodes (no intermediate node sends RREQs), if there does not exist any route for the destination.

```
IF (no route exists)
    check-request buffer for requests already sent for destination
        IF (no request sent already)
            create a RREQ packet
            add (dest addr, broadcast ID) to request buffer
            locally broadcast RREQ
             set timer for RREP WAITTIME before rebroadcasting RREQ
             increment broadcast ID
        ELSE
    buffer packet from stream or discard, according to need
        ENDIF
ENDIF
```

Receiving RREQ

When a node receives a RREQ, it must first of all decide if it already has processed the RREQ. The RREQ is discarded if it has been processed. Otherwise the source address and the broadcastID from RREQ will be buffered to prevent it from being processed again.

```
IF ((source addr, broadcast ID) in request buffer)
    discard request - already heard and processed
ELSE
    add (source addr, broadcast ID) to request buffer
ENDIF
```

The next step is to create or update the route entry in the routing table. This route can be used by the RREP when a route is found.

```
IF (no route to source)
   create a route entry for source addr
ELSE IF (source seqno in RREQ>source seqno in route entry)
   update route entry for source addr
ELSE IF ((source seqno in RREQ = source seqno in route entry) AND (hopcount in RREQ<hop count in route entry))
   update route entry for source addr
ENDIF
```

Then, the node must check if it knows the route to the wanted destination. If the node knows the route, it will unicast a RREP to the source. Otherwise it will forward the RREQ.

```
IF (you are destination of RREQ)
   create a RREP packet
   unicast RREP to source of request
ELSE IF ((have route to destination) AND (destination seqno inroute entry>= destination seqno in RREQ))
  create a RREP packet; unicast RREP to source of request
ELSE
  forward RREEQ
ENDIF
```

Forwarding RREQ

When a node receiving a RREQ that has not processed yet does not have a route, it will forward the RREQ. It creates a RREQ packet first. Copies all fields from received RREQ into new packet, increment hop count field locally broadcast new RREQ packet and discard received RREQ

Forwarding RREP

When a node receives a RREP that is not addressed for the node, it will setup forward route by updating the table and forward the RREP back to the requesting source.

```
IF (route to requested destination does not exist)
    create a route entry for requested destination
ELSE IF (destination seqno in RREP>destination seqno in route entry)
    update-route entry for requested destination
ELSE IF ((destination seqno in RREP = destination seqno in route entry) AND (hop count in RREP<hop count in entry))
    update route entry for requested destination
    IF (route to requesting source exists)
        forward RREP to requesting source
    ENDIF
ENDIF
```

Receiving RREP

When the originating source receives the RREP it will update the routing table.

```
IF (route to destination does not exist)
    create a route entry for destination
ELSE IF (destination seqno in RREP>destination seqno in route entry)
    update route entry for destination
ELSE IF ((destination seqno in RREP = destination seqno in route entry) AND (hop count in RREP<hop count in entry))
    update route entry for destination
ELSE
    discard RREP
ENDIF
```

IV. APPROACH FOR SELFISH NODES

Selfish nodes affect the route in two ways. First, the packet forwarding function performed in the selfish node is disabled for all packets that have a source address or a destination address different from the current selfish node. However, selfish node participates in the Route discovery and Route Maintenance phases of the on-demand protocol. Second, selfish nodes do not participate in the

Route Discovery phase of the reactive protocol. The impact of this model on the network maintenance and operation is more significant than the first one. A selfish node of this type uses the node energy only for its own communications.

Detection of selfish nodes

Selfish node can do the following possible actions in Ad hoc network:

- Does not re-broadcast Route Request (RREQ) when it receives a RREQ.
- Re-broadcasts RREQ but does not forward Route Reply (RREP)on reverse route, therefore the source does not know a route to the destination and it has to rebroadcast a RREQ.
- Re-broadcasts RREQ, forward RREP on reverse route but does not forward data packets.
- Selectively drop data packets.

V. CHANGES MADE TO EXISTING PROTOCOL

In this, it is essential for the detection of this attack to place the participating nodes in promiscuous mode, hence becoming able to overhear the forwarded traffic. Since AODV does not operate in promiscuous mode by default, some modifications had to be performed in the internal files of GloMoSim. First of all, a selfish node has to be induced in the network. For that the structure of selfish node holds the following information.

```
typedef struct SELFISH {
        NODE ADDR destAddr
         BOOL reply;
         BOOL unexpected;
         BOOL drop;
         double time;
}
```

Where, destAddr is the IP address of the node to which the routing traffic was forwarded. Reply is the boolean value that becomes true whenever the offending node replies to a RREQ packet that was forwarded to it. Unexpected is the boolean value that becomes true if the node does not respond as expected to the forwarded traffic. Drop is a boolean value that becomes true whenever we decide that the offending node performs the dropping routing packets attack. Time is a double variable that keeps the time where the offending node was added in the data structure.

Hence, whenever a node forwards routing traffic for which a neighbouring node is not the destination it adds each neighbouring node to the data structure and waits to observe their behaviour. Then if it overhears that a neighbouring node has replied to the forwarded RREQ, it means that it has acted appropriately and it can be removed from the monitoring list. If this is not the case and the packet was a RREP then the offending node has to forward the packet. If it fails to do so within the pre alarm time threshold time period, which was determined by experiments to be seconds, the pre alarm state becomes true. This remains in the pre alarm state for, seconds which is the alarm threshold time period. If the offending node fails to forward the routing packet within this time limit, it moves to the Alarm state. In case of an alarm the legitimate node marks this node as malicious and stops forwarding traffic to it for seconds and it also sends a RERR message to all its upstream neighbours to inform them that all the routes that include this node are not valid. If the received packet is a data packet, normally AODV protocol sends it to the destination address, but behaving as a selfish, it drops all data packets as long as the packet does not come to itself. Performance Metrics are compared after adding selfish node [7].

VI. SIMULATION AND RESULTS

Packet Delivery Ratio(PDR) is defined as the ratio of total number of packets that have reached the destination node to the total number of packets created at the source node [8]. The larger this metric, the more efficient MANET will be.

$$PDR\ \% = \frac{\sum \quad Packets\ received\ by\ destination * 100}{\sum \quad Packet\ sent\ by\ source}$$

End-to-End Delay: It is defined as time taken for a packet to be transmitted across network from source to destination [9]. The metric should have lower value for the efficient network.

$$End - to - End\ Delay = \frac{Sum\ of\ delays\ of\ each\ CBR\ packet\ received}{Number\ of\ CBR\ packet\ received}$$

For Simulation, first AODV was implemented when there was no selfish node. The results are collected from glomo.stat file. Next, selfish nodes were injected in the adhoc network, and selfishness attack on AODV protocol was implemented. In Figure 3, one node is selfish node in SELFISHAODV. It shows that when number of nodes increases and when one node is selfish, then PDR is decreased compared to AODV. When the number of nodes increases and when one node is selfish, then End-to-end Delay is increased compared to AODV.

```
Node: 1, Layer:        MacCSMA, Number of UNICAST packets received clearly: 1250
Node: 1, Layer:    AppCbrServer, (0) Total number of bytes received: 640000
Node: 1, Layer:    AppCbrServer, (0) Total number of packets received: 1250
Node: 2, Layer:        MacCSMA, Number of UNICAST packets output to the channel: 1250
Node: 2, Layer:    AppCbrClient, (0) Total number of bytes sent: 640000
Node: 2, Layer:    AppCbrClient, (0) Total number of packets sent: 1250
```

Fig. 2 screenshot of output collected from glomo.stat file

Table 1 Simulation Parameters

Property	Value
Nodes	10,20,30,40,50,60,70,80
Simulation Time	500 sec
Mobility Model	Random way point
Coverage Area	750m*750m
Maximum Speed	20m/s
Pause Time	1.S
Traffic Type	Constant Bit Rate (CBR/UDP)
Send Rate	10 packets/sec
Packet Size	512 bytes

TABLE II
COMPARISON OF PDR AND END-TO-END DELAY IN AODV AND SELFISHAODV

Nodes	PDR(AODV)	PDR(SELFISHAODV)	End-To-End Delay (AODV)	End-To-End Delay(SELFISHAODV)
10	68.32%	41.36%	201.34ms	564.88ms
20	71.36%	39.40%	588.13ms	1188.61ms
30	47.48%	39.38%	1564.92ms	2494.90ms
40	36.57%	30.83%	1117.01ms	3917.81ms
50	26.76%	26.27%	1568.10ms	7456.03ms
60	23.37%	22.80%	1835.94ms	9353.84ms
70	22.80%	7.26%	982.051ms	7978.32ms
80	8.13%	7.78%	1439.62ms	4019.89ms

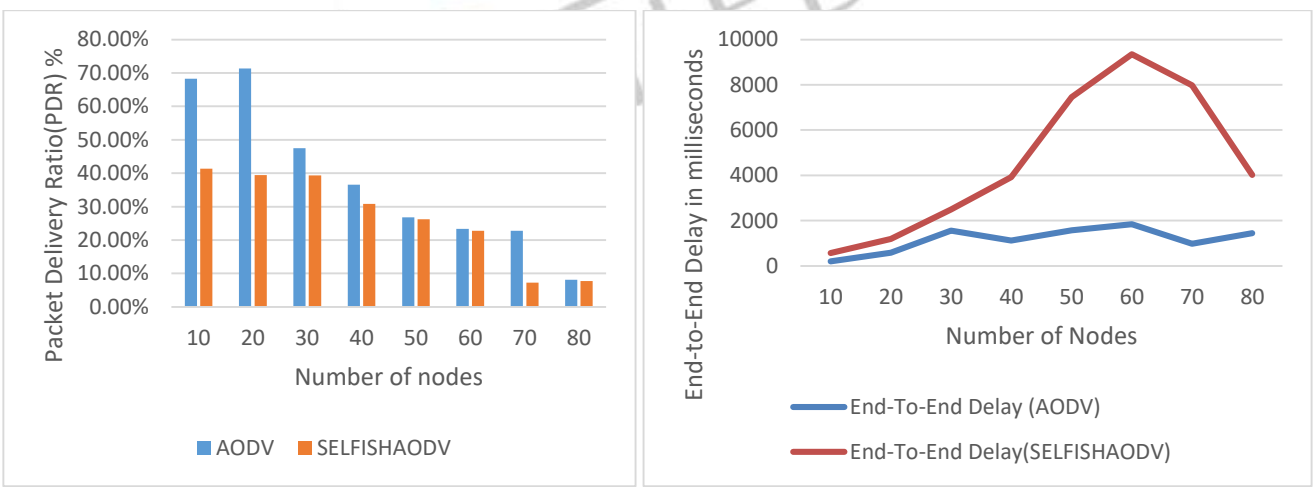

Fig. 1 Comparison of End-to-End Delay and Packet Delivery Ratio

VII. CONCLUSION

Thus, in Mobile ad hoc network, the selfish node does not forward the packets of other nodes and decreases the performance of network as we compare the Packet Delivery Ratio and End-to-End delay with and without presence of the selfish node in the

network. In dense network or normal network, the legitimated node may have accused wrongly by other node due to network error or a selfishness of other node.

REFERENCES

[1] David B. Johnson A. P. Yih-Chun Hu. Sead: secure efficient distance vector routing for mobile wireless ad hoc networks. 2003.

[2] E. M. Royer. A review of current routing protocols for ad hoc mobile wireless networks. April 1999.

[3] Shivahare, Basu Dev, Charu Wahi, and Shalini Shivhare. "Comparison of proactive and reactive routing protocols in mobile adhoc network using routing protocol property." International Journal of Emerging Technology and Advanced Engineering 2.3 (2012): 356-359. K. Elissa.

[4] Djahel, Soufiene, Farid Nait-Abdesselam, and Zonghua Zhang. "Mitigating packet dropping problem in mobile ad hoc networks: proposals and challenges." IEEE communications surveys & tutorials 13.4 (2011): 658-672.

[5] Iqbal, Muddesar, et al. "Design and analysis of a novel hybrid wireless mesh network routing protocol." International Journal of Adaptive, Resilient and Autonomic Systems (IJARAS) 5.3 (2014): 20-39. M. Young, The Technical Writer's Handbook. Mill Valley, CA: University Science, 1989.

[6] Perkins, Charles, Elizabeth Belding-Royer, and Samir Das. Ad hoc on-demand distance vector (AODV) routing. No. RFC 3561. 2003.

[7] Sundararajan, T. V. P., and A. Shanmugam. "Performance Analysis of Selfish Node Aware Routing Protocol for Mobile Ad Hoc Networks." The International Congress for global Science and Technology. 2009.

[8] Subramaniyan, Senthilkumar, William Johnson, and Karthikeyan Subramaniyan. "A distributed framework for detecting selfish nodes in MANET using Record-and Trust-Based Detection (RTBD) technique." EURASIP Journal on Wireless Communications and Networking 2014.1 (2014): 205.

[9] Li, Yong, et al. "The impact of node selfishness on multicasting in delay tolerant networks." IEEE Transactions on Vehicular Technology 60.5 (2011): 2224-2238.