Modified AODV Routing Protocol through Cache Memory for Finding New Routing Paths in MANETs

Swati Gupta¹, Garima Chaudhary²
¹M.Tech. Scholar, RPIIT, Karnal, Haryana (India) swati.mangla.555@gmail.com
²Asstt. Professor, RPIIT, Karnal, Haryana (India) er.garima.chaudhary@gmail.com

Abstract
An ad-hoc network is a multi-hop wireless network where all nodes cooperatively maintain network connectivity without a centralized infrastructure. If these nodes change their positions dynamically, it is called a mobile ad-hoc network (MANET). Since the network topology changes frequently, efficient adaptive routing protocols such as AODV, DSR are used. As the network is wireless, security becomes the major issue in Mobile Ad hoc Networks. Some of the attacks such as modification, fabrication, impersonation and denial of service attacks are due to misbehavior of malicious nodes, which disrupts the transmission. In this paper we proposed an efficient secure AODV routing protocol. Simulation results show that our proposed routing algorithm provides a better level of security and performance than existing works. The simulation results show the improvement of the network performance, in terms of overhead, and end to end delay to the secure AODV routing protocol.

Keywords: MANET, AODV, Secure AODV, Routing Protocol.

1. Introduction to AODV

1.1 Ad hoc On Demand Distance Vector Routing (AODV)
AODV routing is a routing protocol for mobile ad hoc networks and other wireless ad-hoc networks. It is jointly developed in Nokia Research Centre of University of California, Santa Barbara and University of Cincinnati by C. Perkins and S. Das. It is an on-demand and distance-vector routing protocol, meaning that a route is established by AODV from a destination only on demand.

AODV is capable of both uni cast and multicast routing. It keeps these routes as long as they are desirable by the sources. Additionally, AODV creates trees which connect multicast group members. The trees are composed of the group members and the nodes needed to connect the members. The sequence numbers are used by AODV to ensure the freshness of routes. It is loop-free, self-starting, and scales to large numbers of mobile nodes. AODV defines three types of control messages for route maintenance:

RREQ- A route request message is transmitted by a node requiring a route to a node. As an optimization AODV uses an expanding ring technique when flooding these messages. Every RREQ carries a time to live (TTL) value that states for how many hops this message should be forwarded. This value is set to a predefined value at the first transmission and increased at retransmissions. Retransmissions occur if no replies are received.

Data packets waiting to be transmitted (i.e. the packets that initiated the RREQ). Every node maintains two separate counters: a node sequence number and a broadcast_id. The RREQ contains the following fields.

Table 1 RREQ Fields

<table>
<thead>
<tr>
<th>Source address</th>
<th>Broadcast ID</th>
<th>Source sequence no.</th>
<th>Destination address</th>
<th>Destination sequence no.</th>
<th>Hop count</th>
</tr>
</thead>
</table>

IJCMS A
The pair <source address, broadcast ID> uniquely identifies a RREQ. Broadcast_id is incremented whenever the source issues a new RREQ.

RREP- A route reply message is unicast back to the originator of a RREQ if the receiver is either the node using the requested address, or it has a valid route to the requested address. The reason one can unicast the message back, is that every route forwarding a RREQ caches a route back to the originator.

RERR- Nodes monitor the link status of next hops in active routes. When a link breakage in an active route is detected, a RERR message is used to notify other nodes of the loss of the link. In order to enable this reporting mechanism, each node keeps a "precursor list", containing the IP address for each its neighbors that are likely to use it as a next hop towards each destination.

Figure 1.1 A possible path for a route replies if a wishes to find a route to J

The above Figure 1.1 illustrates an AODV route lookup session. Node A wants to initiate traffic to node J for which it has no route. A transmit of a RREQ has been done, which is flooded to all nodes in the network. When this request is forwarded to J from H, J generates a RREP. This RREP is then unicast back to A using the cached entries in nodes H, G and D. AODV builds routes using a route request/route reply query cycle. When a source node desires a route to a destination for which it does not already have a route, it broadcasts a route request (RREQ) packet across the network. Nodes receiving this packet update their information for the source node and set up backwards pointers to the source node in the route tables. In addition to the source node's IP address, current sequence number, and broadcast ID, the RREQ also contains the most recent sequence number for the destination of which the source node is aware. A node getting the RREQ may send a route reply (RREP) if it is either the destination or if it has a route to the destination with corresponding sequence number greater than or equal to that contained in the RREQ. If this is the case, it unicast a RREP back to the source.

Otherwise, it rebroadcasts the RREQ. Nodes keep track of the RREQ's source IP address and broadcast ID. If they receive a RREQ which they have already processed, they discard the RREQ and do not forward it. As the RREP propagates back to the source, nodes set up forward pointers to the destination. Once the source node receives the RREP, it may begin to forward data packets to the destination. If the source later receives a RREP containing a greater sequence number or contains the same sequence number with a smaller hop count, it may update its routing information for that destination and begin using the better route.

As long as the route remains active, it will continue to be maintained. A route is considered active as long as there are data packets periodically travelling from the source to the destination along that path. Once the source stops sending data packets, the links will time out and eventually be deleted from the intermediate node routing tables. If a link break occurs while the route is active, the node upstream of the break propagates a route error (RERR) message to the source node to inform it of the now unreachable destinations. After receiving the RERR, if the source node still desires the route, it can reinitiate route discovery.

Multicast routes are set up in a similar manner. A node wishing to join a multicast group broadcasts a RREQ with the destination IP address set to that of the multicast group and with the 'J'(join) flag set to indicate that it would like to join the group. Any node receiving this RREQ that is a member of the multicast tree that has a fresh enough sequence number for the multicast group may send a RREP. As the RREP's propagate back to the source, the nodes forwarding the message set up pointers in their multicast route tables. As the source node receives the RREPs, it keeps track of the route with the freshest sequence number, and beyond that the smallest hop count to the next multicast group member.
After the specified discovery period, the source nodes will unicast a Multicast Activation (MACT) message to its selected next hop. This message serves the purpose of activating the route. A node that does not receive this message will timeout and delete the pointer. If the node receiving the MACT was not already a part of the multicast tree, it will also have been keeping track of the best route from the RREPs it received. Hence it must also unicast a MACT to its next hop, and so on until a node that was previously a member of the multicast tree is reached.

AODV maintains routes for as long as the route is active. This includes maintaining a multicast tree for the life of the multicast group. Because the network nodes are mobile, it is likely that many link breakages along a route will occur during the lifetime of that route. The counting to infinity problem is avoided by AODV from the classical distance vector algorithm by using sequence numbers for every route. The counting to infinity problem is the situation where nodes update each other in a loop.

Consider nodes A, B, C and D making up a MANET as illustrated in Figure 2.2. A is not updated on the fact that its route to D via C is broken. This means that A has a registered route, with a metric of 2, to D. C has registered that the link to D is down, so once node B is updated on the link breakage between C and D, it will calculate the shortest path to D to be via A using a metric of 3. C receives information that B can reach D in 3 hops and updates its metric to 4 hops. A then registers an update in hop-count for its route to D via C and updates the metric to 5. So they continue to increment the metric in a loop. B will then discard the route and C will be the node with the most recent routing information by which B will update its routing table.

2. About The Simulator

Ns-2 is an object-oriented simulator developed as part of the VINT project at the University of California in Berkeley. The project is funded by DARPA in collaboration with XEROX Palo Alto Research Center (PARC) and Lawrence Berkeley National Laboratory (LBNL). Ns-2 is extensively used by the networking research community. It provides substantial support for simulation of TCP, routing, multicast protocols over wired and wireless (local and satellite) networks, etc.

The simulator is event-driven and runs in a non-realtime fashion. It consists of C++ core methods and uses Tcl and Object Tcl shell as interface allowing the input file (simulation script) to describe the model to simulate. Users can define arbitrary network topologies composed of nodes, routers, links and shared media. A rich set of protocol objects can then be attached to nodes, usually as agents. It had already become the "de facto" standard in networking research.[1, 2]

Although NS is fairly to use once you get to know the simulator, it is quite difficult for a first time user, because there are few user-friendly manuals. Even though there is lots documentation written by the developers who has in depth explanation of simulator, it is written with the depth of a skilled NS user. There already are many user friendly tutorials to make how to use ns-2 easily [3, 4]. They give a new user some basic idea of how the simulator works, how to setup simulation networks, where to look for further information about network components in simulator codes, how to create new network components, and so on. They are mainly focus on simple examples and brief explanations. Although all the usage of the simulator or possible network simulation setups may not be covered in those projects, the projects also help a new user to get quick started. However, with this good tutorials material available today, there still lack documents to ease the analysis work of post-simulation.
Since ns-2 is an academic project, the main purpose is for evaluating the existing network’s performance or the performance of network with new design of component [6, 7]. Thus, to make progress on this problem, this paper is to let people who use ns-2 can easily do the work of network post simulation analysis, which will further make ns-2 more practical on doing scalable simulation.

In the rest of this paper, section 2 will discuss how to conduct a wireless simulation in ns-2; section 3 will describe about the trace file format of ns-2, and then section 4 will do analysis on two sample routing protocols, AODV and DSR, and also show how our performance evaluation mode can work on the analysis of them.

2.1. Software structure and mechanism of NS-2

The key to get to know ns-2 is it is a discrete event network simulator. In ns-2 network physical activities are translated to events, events are queued and processed in the order of their scheduled occurrences. And the simulation time progresses with the events processed. And also the simulation “time” may not be the real life time as we “inputted”. But, why is ns-2 that useful, what kind of work can be done by ns-2, it can model essential network components, traffic models and applications.

Typically, it can configure transport layer protocols, routing protocols, interface queues, and also link layer mechanisms. We can easily see that this software tool in fact could provide us a whole view of the network construction, meanwhile, it also maintain the flexibility for us to decide. Thus, just this one software can help us simulate nearly all parts of the network [1-5]. This definitely will save us great amount of cost invested on network constructing. The following Figure 2.2.1 shows a layered structure which ns-2 can simulate for us.

![Figure 2.1](image1.png)

After the simulation finish, the way ns-2 used to present the most details information on that much network layer is that it provides us a huge trace file recording all the events line by line in it. So, now we see why event driven mechanism is used in ns-2, since it really could maintain the things ever happened as records. And we can trace these records to evaluate the performance of special stuffs in our network, such as routing protocol, Mac layer load, and so on.

![Figure 2.2 data flow for one time simulation](image2.png)

As Figure 2.2 shows, for the data flow of one time simulation in ns-2, the user input an OTcl source file, the OTcl script do the work of initiates an event scheduler, sets up the network topology using the
network objects and the plumbing functions in the library, and tells traffic sources when to start and stop transmitting packets through the event scheduler. And then, this OTcl script file will be passed to ns-2, in this view, we can treat ns-2 as Object-oriented Tcl (OTcl) script interpreter that has a simulation event scheduler and network component object libraries, and network setup module libraries. And then the detail network construction and traffic simulation will be actually done in ns-2. After a simulation is finished, NS produces one or more text-based output files that contain detailed simulation data, and the data can be used for simulation analysis [3, 5].

![Diagram](Image)

**Figure 2.3 Layered structure from the ns-2 developer view**

From the NS-2 developer view, Figure 2.3 shows the layered architecture of ns. The event schedulers and most of the network components are implemented in C++ and available to Tcl Script, thus the lowest level of NS-2 is implemented by C++, and the Tcl script level is on top of it to make simulation stuffs much easier to be conducted. Then, upon the Tcl level, we see the overview of the network. That is the simulation scenario. These all things combined as so called ns-2 software.

### 2.2. Parts needed by one simulation in NS-2

To successfully carry out one simulation, we must first tell ns-2 things it may need from us for one simulation. So what we need is the follow three necessary items:

1) Appearance of the network: the whole topology view of sensor network or mobile network, this includes the position of nodes with (x, y, z) coordinate, the node movement parameters, the movement starting time, the movement is to what direction, and the node movement speed with pausing time between two supposed movement.

2) Internal of the network: Since the simulation is on the network traffic, so it is important we tell the ns2 about which nodes are the sources, how about the connections, what kind of connection we want to use.

3) Configuration of the layered structure of each node in the network, this includes the detail configuration of network components on sensor node, and also we need to drive the simulation, so we need to give out where to give out the simulation results which is the trace file, and how to organize a simulation process.

### 3. Proposition of Advanced AODV

(1) **Existing Work**

AODV is a relative of the Bellmann-Ford distant vector algorithm, but is adapted to work in a mobile environment. Each AODV router is essentially a state machine that processes incoming requests from the network entity. AODV determines a route to a destination only when a node wants to send a packet to that destination. Routes are maintained as long as they are needed by the source. Sequence numbers ensure the freshness of routes and guarantee the loop-free routing. 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:

1) Destination address
2) Next hop address
3) Destination sequence number
4) 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 following flow chart illustrates this process:
From the above mentioned the flow chart implementation, we conclude, aodv is based upon as follows:

We have analysed that mostly nodes in the case of AODV Routing protocol can be idle because of route unavailability for the data transfer between some nodes in the network; Therefore, It has become the need of the hour that we propose the modified AODV protocol whose flow chart is given as follows:

(B) Proposed Work

To improve the idle behaviour of nodes in mobile ad hoc networks due to link breakage, We Introduced as well as exploit the Cache memory to find the alternate and successful route for sending the data from one node to another. Our proposed Protocol model, we named it AAODV, first checks the cache memory to find the successful and possible route between two nodes, in which, successful routes generally stored from previous communication which reduces our time to find the route between nodes every time when they require. Also, It checks whether the stored route in cache possess the property of link breakage. In that case, We apply another strategy to repair the broken route and if route is unable to be repaired by the algorithm, then,
It will start to search another possible and shortest route between the two nodes.

And if the Proposed Protocol does not find a suitable route from cache memory, only then, it will start a new route as AODV does.

4. Experimental Comparison between AODV and AAODV:

We have already showed the separate performance graph metrics of AODV and AAODV respectively. Now, we are showing the performance of two protocols on single metric (pink color is for AODV protocol and green is for AAODV) one by one which is as follows:

1. Total Packets received

   ![Total no of packets received using AODV and AAODV](image)

   Figure 3.3 Total no of packets received using AODV and AAODV

   In the case of Total no. of packets received, we can easily analyze that AAODV receives around 2900 packets whereas AODV receives only 2500 packets when 3000 packets were sent in both cases which shows the betterment in AAODV comparing to AODV.

2. Total Packets lost

   ![Total no of packets lost using AODV and AAODV](image)

   Figure 3.4 Total no of packets lost using AODV and AAODV

   In the above, Packet lost is 100 in case of AAODV and 500 is for AODV which makes AAODV once again better than AODV protocol.

3. Total Time delay

   ![Total time delay using AODV and AAODV](image)

   Figure 3.5 Total time delay using AODV and AAODV

   In the above, Packet lost is 100 in case of AAODV and 500 is for AODV which makes AAODV once again better than AODV protocol.
In the above, Total Time Delay is approximately 3.3 in case of AODV and 1.1 for AAODV which conclude once again the fact that AAODV is better than AODV.

5. Conclusion and Future Work

A new model has been proposed which takes care of route maintenance feature based on priority of route for communication from source to destination. The new scheme will be incorporated on AODV and the results will be compared with existing AODV extensions with embedded route discovery and route maintenance concept. Almost results are expected to be better than the standard AODV. The graphical notations are used for representation. The metrics used are Packet receiving ratio, Total packet ratio and the time delay for nodes in sending the data. Successful delivery of different route discovery and maintenance message is very important in a MANET as a lot of route discovery effort is wasted if a reply message is lost; moreover a new route discovery process has to be reinitiated. Our simulation results show that AAODV protocol has better network lifetime with minor change in throughput in Wireless Ad hoc network.

In this study the two on-demand routing protocols AODV & AAODV are analyzed and their performances have been evaluated. This thesis can be enhanced by analyzing other MANET routing protocols with different traffic sources.

References