ISSN No: 2348-4845

Implementation of AODV Routing Protocol in NS2: A comparison with DSDV routing protocol

N Dinesh Kumar, Associate Professor & Head - EIE,
Vignan Institute of Technology & Science, Deshmukhi, Andhra Pradesh, India–508284. *dinuhai@yahoo.co.in*

Anuj Saxena, Final year, Electronics and Instrumentation Engineering,
VITS, Deshmukhi, Andhra Pradesh, India. *saxena.anuj28@gmail.com*

Arvind Sundararajan, Final year, Electronics and Instrumentation Engineering,
VITS, Deshmukhi, Andhra Pradesh, India. *asrajan93@gmail.com*

N. Roop Kiran, Final year, Electronics and Instrumentation Engineering,
VITS, Deshmukhi, Andhra Pradesh, India. *roop.kiran92@gmail.com*

In view of the algorithm and the advantages that the AODV routing protocol presents in front of us, we attempt to emulate the same in our project with the help of the software NETWORK SIMULATOR 2. In this paper the AODV routing protocol is evaluated graphically with the help of a tool called NAM (Network Animator) and a statistical analysis is drawn with the help of TRACEGRAPH for a network having 10 nodes. The same analysis is done for the same network employing the DSDV routing protocol and its performance is compared with the performance of the network employing the AODV routing protocol. This project is an attempt to acquire an al-round perspective of the AODV protocol in comparison with the DSDV protocol.

Keywords: *Analysis, AODV, comparisons, DSDV, NS-2, Tracegraph*

Introduction

The limited resources in MANETs have made designing of an efficient and reliable routing strategy a very challenging problem. An intelligent routing strategy is required to efficiently use the limited resources while at the same time being adaptable to the changing network conditions such as: network size, traffic density, network partitioning. In parallel with this, the routing protocol may need to provide different levels of QoS to different types of applications and users. AODV routing protocol is one such protocol, classified as a reactive routing protocol, which caters to all the above mentioned criterias.

AODV ROUTING PROTOCOL

Ad hoc On-Demand Distance Vector (AODV) Routing is a routing protocol for mobile ad hoc networks (MANETs) and other wireless ad hoc networks. It is jointly developed in Nokia Research Center, University of California, Santa Barbara and University of Cincinnati by C. Perkins, E. Belding-Royer and S. Das.

In AODV, the network is silent until a connection is needed. At that point the network node that needs a connection broadcasts a request for connection. Other AODV nodes forward this message, and record the node that they heard it from, creating an explosion of temporary routes back to the needy node. When a node receives such a message and already has a route to the desired node, it sends a message backwards through a temporary route to the requesting node. The needy node then begins using the route that has the least number of hops through other nodes. Unused entries in the routing tables are recycled after a time. When a link fails, a routing error is passed back to a transmitting node, and the process repeats.

Much of the complexity of the protocol is to lower the number of messages to conserve the capacity of the

ISSN No: 2348-4845

network. For example, each request for a route has a sequence number. Nodes use this sequence number so that they do not repeat route requests that they have already passed on. Another such feature is that the route requests have a "time to live" number that limits how many times they can be retransmitted. Another such feature is that if a route request fails, another route request may not be sent until twice as much time has passed as the timeout of the previous route request.

The advantage of AODV is that it creates no extra traffic for communication along existing links. Also, distance vector routing is simple, and doesn't require much memory or calculation. However AODV requires more time to establish a connection, and the initial communication to establish a route is heavier than some other approaches.

DSDV ROUTING PROTOCOL

Destination-Sequenced Distance-Vector Routing (DSDV) is a table-driven routing scheme for ad hoc mobile networks based on the Bellman–Ford algorithm. It was developed by C. Perkins and P.Bhagwat in 1994. The main contribution of the algorithm was to solve the routing loop problem. Each entry in the routing table contains a sequence number, the sequence numbers are generally even if a link is present; else, an odd number is used. The number is generated by the destination, and the emitter needs to send out the next update with this number. Routing information is distributed between nodes by sending *full dumps* infrequently and smaller incremental updates more frequently.

If a router receives new information, then it uses the latest sequence number. If the sequence number is the same as the one already in the table, the route with the better metric is used. Stale entries are those entries that have not been updated for a while. Such entries as well as the routes using those nodes as next hops are deleted.

DSDV requires a regular update of its routing tables, which uses up battery power and a small amount of bandwidth even when the network is idle.Whenever the topology of the network changes, a new sequence number is necessary before the network re-converges; thus, DSDV is not suitable for highly dynamic networks. (As in all distance-vector protocols, this does not perturb traffic in regions of the network that are not concerned by the topology change.)

Wireless Simulation in NS-2

Software Structure and Mechanism in 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. This definitely will save us great amount of cost invested on network constructing. The following Figure 1 shows a layered structure which ns-2 can simulate for us.

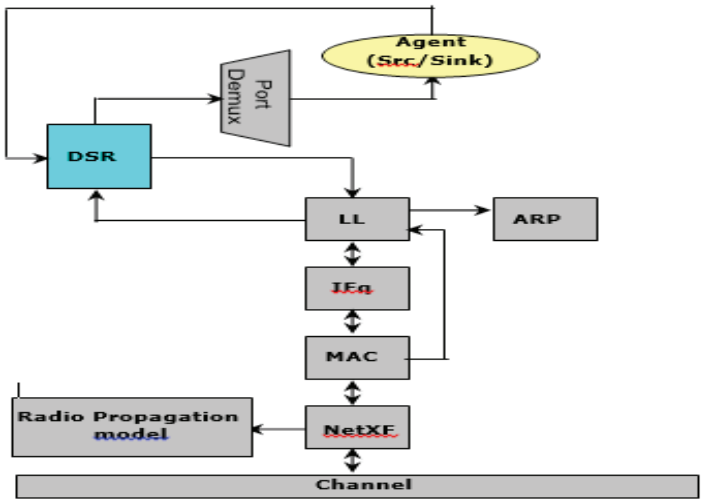

Figure 1. NS-2 simulate layered structure of network

After the simulation finishes, ns-2 presents the detailed information to the network layer by means of a trace file which is a line by line account of all the events. This

ISSN No: 2348-4845

shows the significance of an event driven mechanism which records the events as they occur. These records can then be traced to evaluate the performance of critical things in the network such as routing protocol, MAC layer load, etc.

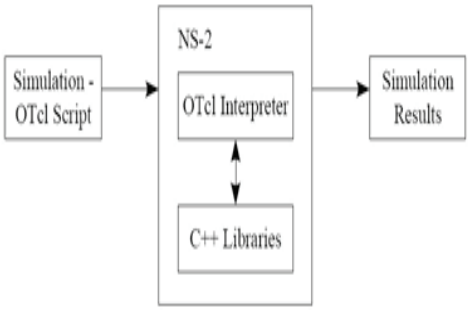

Figure 2. Data flow for 1 time simulation.

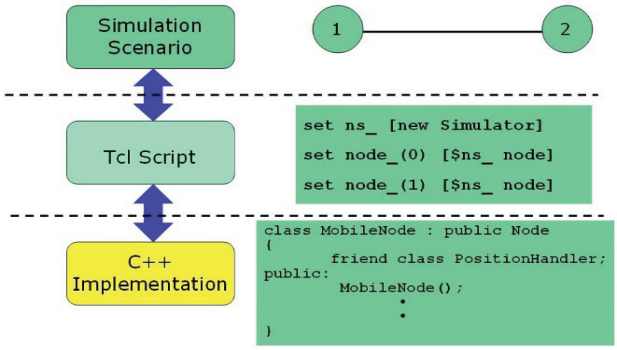

Figure 3. Layered Structure of NS2

Figure 2 shows, the data flow of one time simulation in ns-2, the user inputs an OTcl source file, the OTcl script does the work of initiating 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 is 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 set up 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.

The figure 3 shows the ns-2 developer's view, the layered structure of ns. The event schedulers and the network components are implemented in C++ and is made available to the TCL script, thus the lowest level of ns-2 is implemented in C++, and the TCL script is placed at the top to make simulation easier. We see the overview of the network on top of the TCL level. This defines the simulation scenario. All these things combined is the ns-2 software.

Requirements for running simulation inNS2

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.

2.3. Writing TCL to run a simple wireless simulation

In this section, we then will present step by step of how to do all the things as needed by one simulation in ns-2 with one tcl scripts sequence:

Step 1. Create an instance of the simulator:

Step.2. Setup trace supportby opening file "trace_bbtr.tr" and call the procedure trace-all

Step 3. Create a topology object that keeps track # of all the nodes within boundary

Step 4. The topography is broken up into grids and the

ISSN No: 2348-4845

default value of grid resolution is 1. A different value can be passed as 3rd parameter to load_flatgrid {}.

Step 5. Create the object God, "God (General Operations Director) is the object that is used to store global information about the state of the environment, network or nodes. The procedure create-god is defined in $NS2_HOME/tcl/mobility/com.tcl, which allows only a single global instance of the God object to be created during a simulation. God object is called internally by MAC objects in nodes, so we must create god in every cases.

Step 6. Before we can create node, we first needs to configure them. Node configuration API may consist of defining the type of addressing (flat/hierarchical etc), for example, the type of adhoc routing protocol, Link Layer, MAC layer, IfQ etc.

Step 7. Create nodes and the random-motion for nodes is disabled here, as we are going to provide node position and movement (speed & direction) directives next

Step 8. Give nodes positions to start with, Provide initial (X,Y, for now Z=0) co-ordinates for node_(0) and node_(1) and the rest of the nodes. Nodeo has a starting position of (5,2) while Node1 starts off at location (390,385). The positions are assigned to all the rest of the nodes.

Step 9. Setup node movement as the following example, at time50.0s, node1 starts to move towards the destination (x=25, y=20) at a speed of 15m/s. This API is used to change direction and speed of movement of nodes.

Step 10. Setup traffic flow between the two nodes as follows: TCP connections between node_(0) and node_(1)

Step 11. Define stop time when the simulation ends and tell nodes to reset which actually resets their internal network components. In the following case, at time 150.0s, the simulation shall stop. The nodes are reset at that time and the "$ns_ halt" is called at 150.0002s, a little later after resetting the nodes. The procedure stop{} is called to flush out traces and close the trace file.

Step 12. Finally the command to start the simulation

So, these 12 steps could finish one time simulation

Note: The above mentioned steps can be used to create a wireless network for n number of nodes and the flow of traffic, the packet size, etc can be specified by the type of protocol used (tcp, udp, cbr).

These 12 steps into one tcl file and do the simulation. However, there exist some problems on such kind of use on typical network performance test situations. Performance testing usually needs to be scalable in the number of nodes and network transmitting packets. Suppose for one network there are hundreds of nodes, we need to set all of the nodes' positions and their movement, this a huge amount of workload, also, suppose we need to setup all the possible sources and destinations and even connections, also is a huge workload, Furthermore, even if we can set them, we cannot guarantee our input is randomly selected, which is necessary for a fair comparison.

Performance comparison on AODV & DSDV

The parameters that are considered that makes AODV routing protocol different from its parent protocol ie DSDV are:

- Throughput vs average RTT
- Cumulative sum of dropped packets
- Cumulative sum of the generated packets
- Jitter

Throughput Analysis

Throughput or **network throughput** is the rate of *successful* message delivery over a communication channel. This data may be delivered over a physical or logical link, or pass through a certain network node. The throughput is usually measured in bits per second (bit/s or bps), and sometimes in data packets per second or data packets per time slot.

The **round-trip delay time (RTD)** or **round-trip time (RTT)** is the length of time it takes for a signal to be sent

ISSN No: 2348-4845

plus the length of time it takes for an acknowledgment of that signal to be received. This time delay therefore consists of the propagation times between the two points of a signal.

Throughput vs average RTT

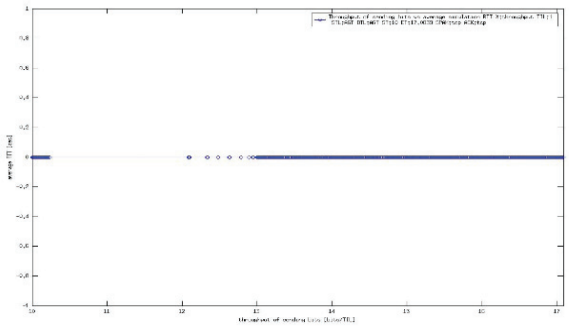

Figure 4a: DSDV

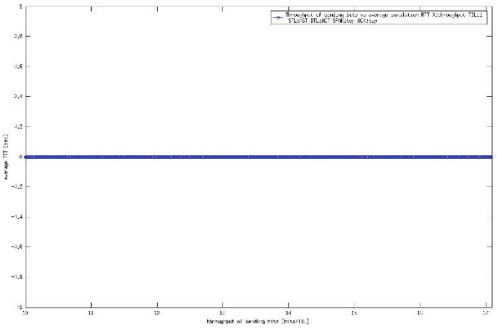

Figure 4b: AODV

Throughput for entire simulation time.

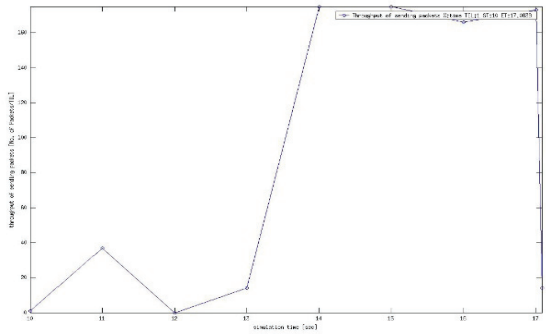

Figure 4c: DSDV

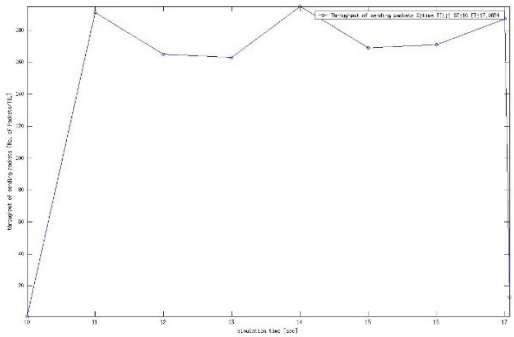

Figure 4d: AODV

On observing the graphs figures 4c and 4d we can see that the throughput of the AODV network is constantly high during the simulation time whereas it is lesser initially and pickups up in the case of DSDV, this is because of the delay caused in searching for the optimum route to the destination by verifying the dynamic vector tables. The other figures 4a and 4b show the continuity of the round trip time for the sent packets with respect to the throughput.

Packet Analysis

Cumulative sum of dropped packets

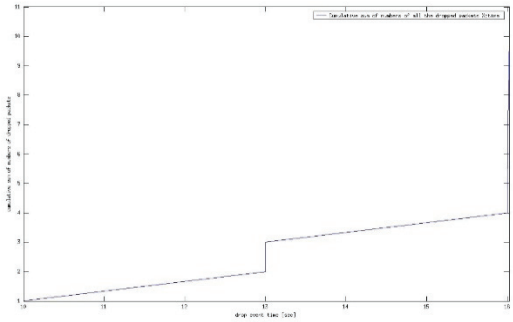

Figure 5a: DSDV

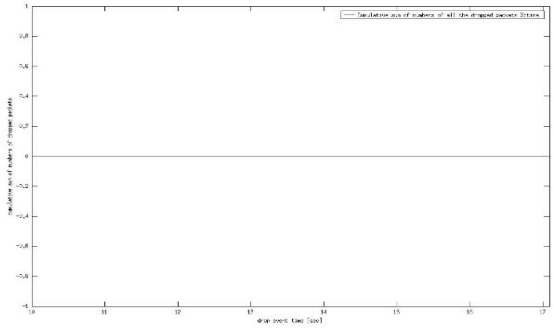

Figure 5b: AODV

ISSN No: 2348-4845

The figures 5a, 5b, 5c, 5d form a part of the packet analysis for the network running on DSDV and AODV routing protocols respectively. The graphs 5a and 5b compare the number of dropped packets in the entire simulation time where it is observed that the number of packets dropped in DSDV routing protocol are more and there are no packets dropped in AODV routing protocol. This shows that AODV routing protocol is efficient than DSDV routing protocol. The packets generated in AODV increase at the outset of communication in AODV whereas the generated packets only increase after a delay in the case of DSDV.3.3. Cumulative Distribution of Jitter

Cumulative sum of generated packets

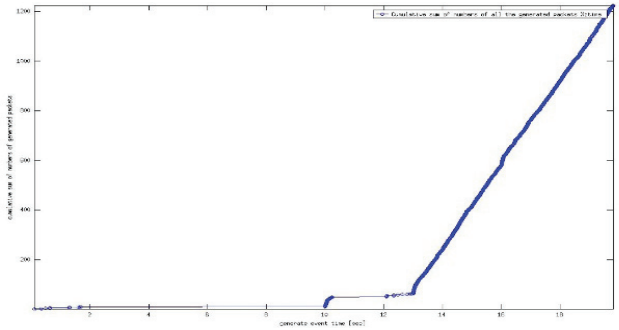

Figure 5c: DSDV

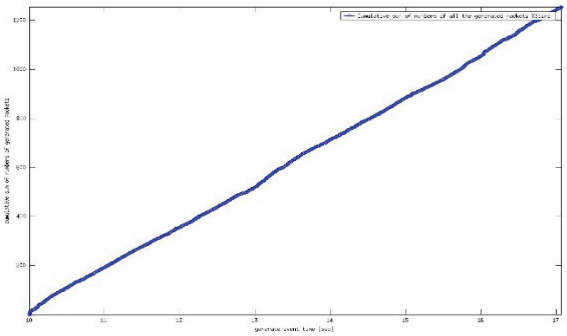

Figure 5d: AODV

The figures 6a and 6b depict the cumulative distribution of jitter over the simulation jitter (sec) for DSDV and AODV respectively. Here we can observe that in both the cases the jitter increases in an exponential manner but the overall jitter stabilizes faster in DSDV when compared to AODV.

Jitter Cumulative Distribution

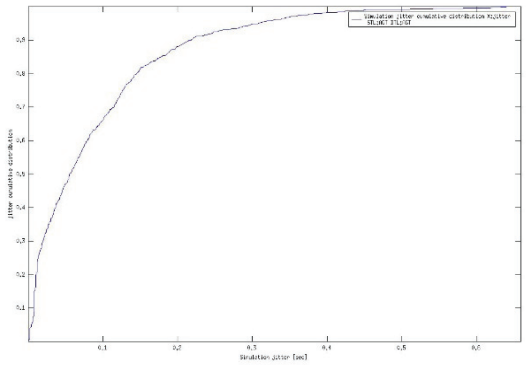

Figure 6a: DSDV

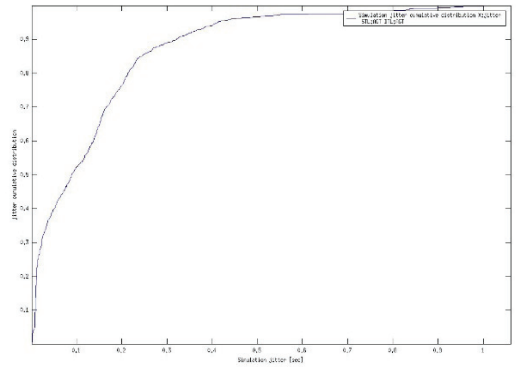

Figure 6b:AODV

Jitter is defined as a variation in the delay of received packets. At the sending side, packets are sent in a continuous stream with the packets spaced evenly apart. Due to network congestion, improper queuing, or configuration errors, this steady stream can become lumpy, or the delay between each packet can vary instead of remaining constant.

Conclusions

The results obtained from the performance analysis made by using NS2 show that these two routing protocols are very much similar in the basic approach and vary slightly in a few metrics such as delay, throughput. Considering the observations made from the graphs suitable conclusion can be drawn which inclines to the fact that AODV protocol is better suited for dynamic networks than the DSDV protocol which uses the traditional table driven approach which in cases of large networks is slower.

References

[1] The network simulator - ns-2. http://www. isi. edu/nsnam/ns/.

[2] K. Fall and K. Varadhan (Eds.), 1999. ns notes and documentation,

http://wwwmash.cs.berkeley.edu/ns/.

[3] NS by Example, http://nile.wpi.edu/NS/

[4] Introduction to Shell Scripting. http://www.uwsg. iu.edu/usail/concepts/shell-scripting.html

[5] Steve's Bourne scripting tutorial. h t t p : / / steveparker.org/sh/sh.shtml

[6] Shell Programming. http://www.linuxfocus.org/ English/September2001/article216.shtml

[7] NS-2 Trace Formats. http://k-lug. org/~griswold/NS2/ns2-trace-formats.html

[8] Andrew S. Tanenbaum, Computer Networks, Fourth Edition,

[9] http://www.cisco.com/c/en/us/support/docs/ voice/voice-quality/18902-jitter-packet-voice.html

[10] Mehran Abolhasan, Tadeusz Wysocki, Eric Dutkiewicz, A review of routing protocols for Mobile Ad Hoc Networks

Paid Inclusion in Search Engine Results

Paid inclusion involves a search engine company charging fees for the inclusion of a website in their results pages. Also known as sponsored listings, paid inclusion products are provided by most search engine companies, the most notable being Google.

The fee structure is both a filter against superfluous submissions and a revenue generator. Typically, the fee covers an annual subscription for one web-page, which will automatically be cataloged on a regular basis. However, some companies are experimenting with non-subscription based fee structures where purchased listings are displayed permanently. A per-click fee may also apply. Each search engine is different. Some sites allow only paid inclusion, although these have had little success. More frequently, many search engines, like Yahoo!, mix paid inclusion (per-page and per-click fee) with results from web crawling. Others, like Google (and as of 2006, Ask.com), do not let webmasters pay to be in their search engine listing (advertisements are shown separately and labeled as such).

Some detractors of paid inclusion allege that it causes searches to return results based more on the economic standing of the interests of a web site, and less on the relevancy of that site to end-users.

Often the line between pay per click advertising and paid inclusion is debatable. Some have lobbied for any paid listings to be labeled as an advertisement, while defenders insist they are not actually ads since the webmasters do not control the content of the listing, its ranking, or even whether it is shown to any users. Another advantage of paid inclusion is that it allows site owners to specify particular schedules for crawling pages. In the general case, one has no control as to when their page will be crawled or added to a search engine index. Paid inclusion proves to be particularly useful for cases where pages are dynamically generated and frequently modified.

Paid inclusion is a search engine marketing method in itself, but also a tool of search engine optimization, since experts and firms can test out different approaches to improving ranking, and see the results often within a couple of days, instead of waiting weeks or months. Knowledge gained this way can be used to optimize other web pages, without paying the search engine company.