A Comparative Study of AODV, DSR, and DYMO routing protocols using OMNeT++

Nitin Kumar[1], Kunj Vashishtha[2], Kishore Babu[3]

[1]Asst. Professor, Department of Electronics, SRM University, NCR Campus, Ghaziabad [U.P] INDIA
[2]Student, B.Tech, Department of Electronics, SRM University, NCR Campus, Ghaziabad [U.P] INDIA
[3]Student, B.Tech, Department of Electronics, SRM University, NCR Campus, Ghaziabad [U.P] INDIA

nitindhama85@gmail.com
kunjvas@hotmail.com
kishore.babu@gmx.com

***Abstract*—The work presented here is a summary of the results obtained when routing protocols viz. AODV, DSR, DYMO were simulated using virtual hosts on a discrete-event simulator: OMNeT++ v4.2.1. The three protocols are run on a simulation setup of 20 nodes without any mobility models. This allows us to focus our attention on solely the MAC properties and related results derived from the three protocols. We describe and compare the three routing protocols on available parameters like contention window, SNIR, routing overhead, radio state and more. We conclude by stating the DYMO emerges as the better protocol of the three examined here.**

***Keywords*—MANET, Routing Protocols, OMNeT++, network simulation, AODV, DSR, DYMO.**

I. Introduction

In our advent into the world of network simulation, we have chosen to taken up a subject, which is in the center of many important developments in the modern world. Wireless communication has seen its boundaries extended with the addition of ad-hoc mode capabilities. It is our intent to contribute, in our own small way, to this growing pool of knowledge.

Using a simulation technique allows one to analyze network protocols large scale – subject to available computing power. There are a huge number of simulation tools available. In this case study, we focus on OMNeT++.

The paper goes on to describe various aspects regarding routing protocols used in mobile ad-hoc networks. Section 2 describes OMNeT++ in its various features and its relevance with the network simulation space. Section 3 lists out the simulation setup used, and describe the different hardware and software parameters of the simulation workbench. Section 4 and 5 analyze the results obtained, while drawing some conclusions. Finally, the paper end with a look at future steps in the direction and list of works referenced which were helpful in guiding us in our work.

II. OMNeT++

A. Introduction

OMNeT++ is a simulation environment free for academic use. The OMNeT++ engine runs discrete, event-driven simulations of communicating nodes on a wide variety of platforms and is becoming increasingly popular in the field of network simulation.

OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. "Network" is meant in a broader sense that includes wired and wireless communication networks, on-chip networks, queuing networks, and so on. Domain-specific functionality such as support for sensor networks, wireless ad-hoc networks, Internet protocols, performance modeling, photonic networks, etc., is provided by model frameworks, developed as independent projects. OMNeT++ offers an Eclipse-based IDE, a graphical runtime environment, and a host of other tools. There are extensions for real-time simulation, network emulation, alternative programming languages (Java, C#), database integration, SystemC integration, and several other functions.

A hierarchy of small represents scenarios in OMNeT++, reusable modules written in C++. OMNeT++ supports behavioral modeling of modules using finite state machines and communication within and between modules is primarily based on message passing, but the foundation in open source software written in C++ also allows for a rapid prototyping approach to module development. Modules' relationships and communication links are stored as plain-text Network Description (NED) files and can be modeled graphically. Simulations are either run interactively in a graphical environment or are executed as command-line applications.

B. INET Framework

The INET Framework extension is a set of simulation modules released under the GPL. It provides OMNeT++ modules that represent various layers of the Internet protocol suite, e.g. the TCP, UDP, IPv4, and ARP protocols. The INET Framework also features models for wireless radio communication including radio distribution models and various MAC protocols such as IEEE 802.11. Mobility modeling was introduced into OMNeT++ with the Mobility Framework. It was later incorporated into the INET Framework for modeling spatial relations of mobile nodes.

INET Framework contains IPv4, IPv6, TCP, SCTP, UDP protocol implementations, and several application models. The framework also includes an MPLS model with RSVP-TE and LDP signaling. Link-layer models are PPP, Ethernet and 802.11. Static routing can be set up using network auto configurators, or one can use routing protocol implementations.

The INET Framework supports wireless and mobile simulations as well. Support for mobility and wireless communication has been derived from the Mobility Framework.

C. INET-MANET

INETMANET is based on INET Framework and is continuously developed. Generally it provides the same functionality as the INET Framework, but contains additionalprotocols and components that are especially useful while modeling wireless communication.

In conclusion, OMNeT++ and the INET Framework provide all the necessary components for simulating Internet protocols in general and MANET protocols in particular. Because of its modular architecture and its ability to directly access, monitor and alter all modules' internal states, OMNeT++ is very well suited for the implementation of complex protocols.

III. Simulation Setup

We chose to execute the simulation of the network on separate machines so as to understand the varying effects of the supporting hardware had on the simulation experience.

TABLE I
The Hardware/Software Setup

Operating System	Apple Mac OS X 10.7.3
Processor	Intel Core 2 Duo - 2.26 GHz
Memory	5GB
Compiler	gcc
Simulation Environment	OMNeT++ 4.2.1
Simulated using	Cmdenv, Tcl/Tkenv

Over the course of the many simulations that led to the final ones presented here, we understood that OMNeT++ is a highly resource intensive simulation package. When a network was simulated using the Tcl/Tkenv graphical environment, it was noticed that the simsec/sec was abysmally low. We attributed this phenomenon to the additional resources necessary to support the graphical environment.

Once we gained enough confidence over our network model, and general proficiency over protocol implementation on our network, we moved to simulate the network on a command-line environment using `Cmdenv`. The run configuration we designed allowed us to use the multiple processors available on our different computers.

`Cmdenv-express-mode` allowed us to reach `simsec/sec` of about 4. This allowed a more efficient approach to record data for the analysis.

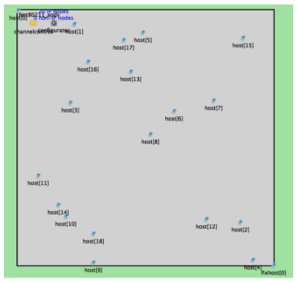
A. Simulation Setup

TABLE 2
The Simulation Setup

Playground Dimensions	1000m X 1000m
No. Of Wireless Hosts	20
Mobility Model	None
Max. Channel Power	2.0mW
Radio Tx Power	2.0mW
Radio Bitrate	54Mbps
Broadcast Delay	0s – 0.005s
Simulation Time	3600s (1 Hour)
Total Packets Sent	17929.0
Time Begin	10s
Message Length	512B
Message Frequency	0.2s
Routing Protocols	AODV, DSR, DYMO
Simulation Style	Cmdenv-express-mode

We must admit that our software configuration with respect to the real world conditions are fairly limiting. We intended to simulate the ad-hoc networks using IEEE 802.11g standard specifications. In order to improve the correlation between this software model and real world applications, we use a message burst length of 512B lasting almost zero seconds, with a frequency of about 0.2 seconds. This enables us to approach an almost brute-force condition on the network so as to stress it out. We use 20 hosts randomly spread out on a simulated playground of 1km^2. We felt should begin without the use of a mobility model for our initial analysis. We believe that our future work (Section 7) shall incorporate a few mobility models over a similarly designed network. Our simulated time is exactly one hour. This relatively large amount of time allowed us to monitor the three protocols over a longer period. In some of the results (Section 5) we noticed that a longer simulation time brought out characteristics of the protocol against one another, that one might not have noticed over a small simulation time.

Figures 1 & 2 are illustrations using a computer screenshot taken during the simulation progress of the network using the Tcl/Tkenv graphical environment. These screenshots were taken on the Mac OS X machine described earlier.



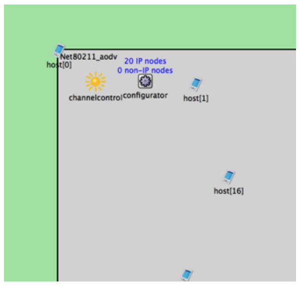
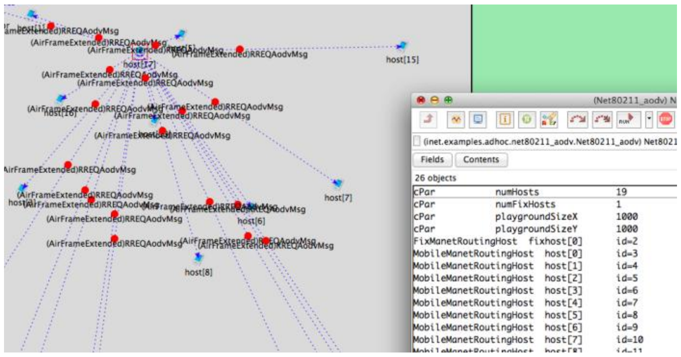


Figure 1a – 1b: The network design illustration showing the `source[host0]` and `destination[fixhost0]` on the top left and bottom right corners of our 'playground'



```
(inet.examples.adhoc.net80211_aodv.Net80211_aodv) Net8021
26 objects
cPar                  numHosts          19
cPar                  numFixHosts       1
cPar                  playgroundSizeX   1000
cPar                  playgroundSizeY   1000
FixManetRoutingHost   fixhost[0]        id=2
MobileManetRoutingHost  host[0]         id=3
MobileManetRoutingHost  host[1]         id=4
MobileManetRoutingHost  host[2]         id=5
MobileManetRoutingHost  host[3]         id=6
MobileManetRoutingHost  host[4]         id=7
MobileManetRoutingHost  host[5]         id=8
MobileManetRoutingHost  host[6]         id=9
MobileManetRoutingHost  host[7]         id=10
```

Figure 2: The network activity shows the AODV overhead packets being broadcasted by `host[17]`. The screenshot also shows the parameter analyser present as a part of the OMNeT++ package.

IV. Results

A few of the parameters that were analysed are presented as follows.

A. Contention Window

A Contention Window (CW) is a parameter, which is used to arbitrate between High priority and low priority traffic, especially where the application of Enhanced Distributed Channel Access (EDCA) is seen. To decide between AODV, DSR, or DYMO on the basis of the contention windows, one must look for a lower value. Figure 3 shows a graph generated by the vector data of the three protocols during simulation in a time-averaged manner. It clearly shows that DYMO has a smaller contention window when compared with AODV and DSR. This is an attribute in support of the DYMO protocol.

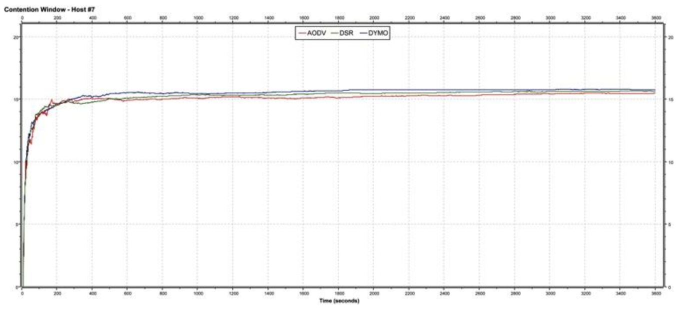


Figure 3: The contention window parameter as measured at Host #7, located at an intermediary position between the source and destination hosts.

B. Signal-to-Noise plus Interference Ratio (SNIR)

Signal-to-noise plus interference (SNIR) is defined as the ratio of signal power to the combined noise and interference power. Figure 4 shows the time-averaged values for SNIR at an intermediary host #9. It is understood, that a higher SNIR value is beneficial, for that would mean larger signal power vis-à-vis the noise and interference power. Figure 4 shows almost comparable characteristics for each of the three protocols. But, this is one of the scenarios where a longer simulation time has shown its importance. Initially where AODV and DSR dominate DYMO with higher numbers, it is seen that as time passes by, DYMO is seen to rise higher than other two protocols in question. There is no clear winner between AODV and DSR, but DYMO keeps itself a notch higher than the two.

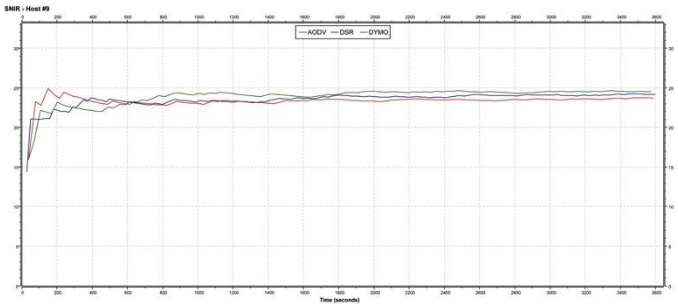


Figure 4: SNIR as measured at Host #9, located at an intermediary position between the source and destination hosts.

C. Radio State

Radio State can be understood as the instantaneous transmitter (tx) power consumed at a certain wireless radio host. This parameter has been measured on a time-averaged manner at two hosts: the source and the destination hosts of our network. Figure 5 and 6 show the time-averaged graphs for destination host and source host respectively. At a glance, the reader is able to deduce that DYMO consumer the lowest power when compared with AODV and DSR. This remains true for either case: the source and the destination hosts. Hence, again DYMO continues to prove its supremacy over protocols like AODV and DSR. The radio state parameter has demonstrated a relatively large, and significant, advantage DYMO presents to network designers.

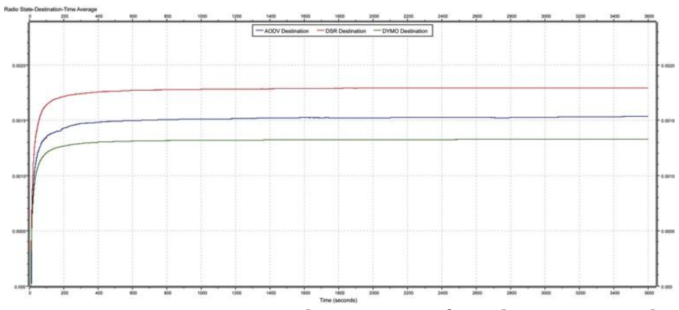


Figure 5: A time averaged measure of Radio State at the destination host.

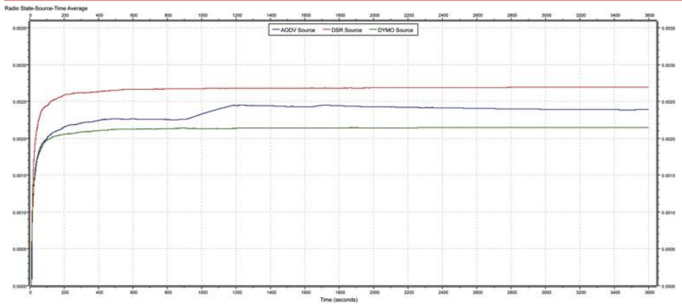


Figure 6: A time averaged measure of Radio State at the source host.

D.MAC Loss Rate

The MAC loss rate is calculated as a ratio of packets lost at the MAC layer. This is an important parameter when packet loss is not acceptable in certain kind of networks. Figure 7 presents the case for the MAC loss rates for the 3 protocols over the same network (time-averaged).

A lower MAC loss rate is desirable. A simple understanding of Figure 7 shows DYMO has the lowest MAC loss rate. While one must note that AODV may begin with a higher loss rate, it tends to perform better with some time after the network is active. This may indicate the feature of AODV where it understands the topology of the network better once its routing overhead is in place.

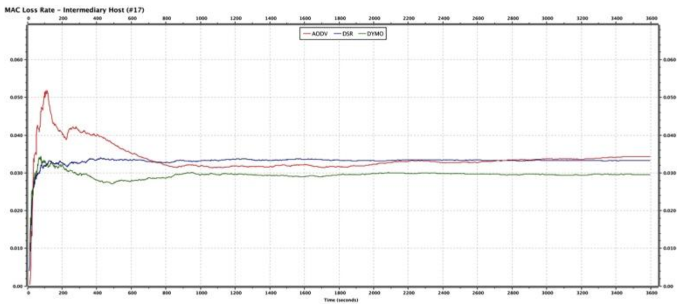


Figure 7: MAC Loss Rate measured, in a time averaged manner over the simulated time, measured at Host #17, located at an intermediary position between the source and destination hosts.

E. Packets received in queue

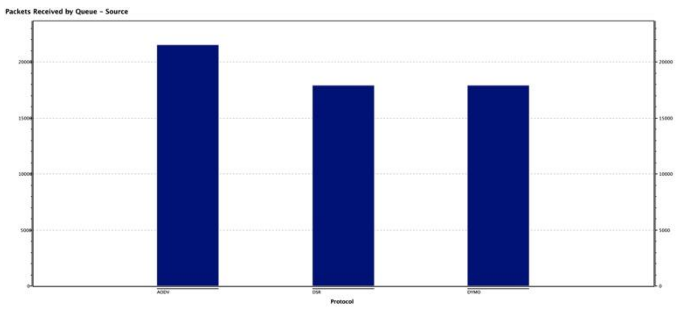


Figure 8: A Bar Chart depicting the Packets received by a MAC queue at the `source`.

The number of packets received by the queue at the source can be used to understand how quickly can a packet be sent out to the network for routing. A longer queue indicates a correspondingly longer delay with which a packet is ready for the routing process. This parameter is especially critical for reactive protocols because of the lack of a pre-existing routing table mechanism in the hosts and network.

Figure 8 depicts the different queue lengths recorded at the source host under the three protocols. AODV had the longest queue while DSR and DYMO had comparable lengths of their respective queues.

F. Routing Overhead

A routing overhead can be seen as a price to pay to route a packet through the network. Another description of a routing overhead would be to understand it as the maintenance load on the network, used for the regular upkeep of the routes present on the network.

Figure 9 shows routing overheads for AODV and DYMO. Not surprisingly, DYMO has about a thousand times smaller footprint on the network for its maintenance. AODV, due to its inherent principles, requires a rather large routing overhead.

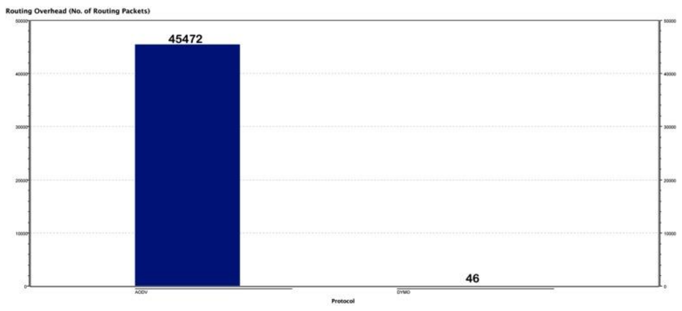


Figure 9: Routing overhead – AODV vs DYMO.

G.Packet Collisions

The idea of a packet collision is self-explanatory. When two or more packets cross each others path at the same time, it causes mutual interference which may render one or either of the packets unusable by the intended physical layer. The lower the number of packet collisions the better it is for the network.

Figure 10 shows statistics related to the packet collisions in the network on all three protocols. DSR has the highest number of packet collisions, while AODV being lower. However DYMO has the lowest number of packet collisions.

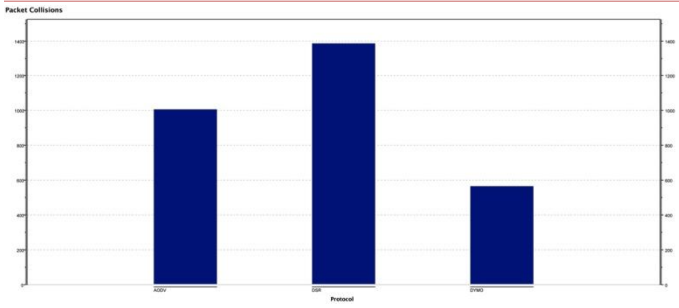


Figure 10: Packet Collisions recorded.

That concludes the different parameters we wished to discuss as a part of this paper. We believe that the presented parameters are critical in the evaluation and discussion of any set of routing protocols in a ad-hoc setup. We shall now go on to draw out some conclusions on the next section.

V.CONCLUSION

In the previous section, we have compared and evaluated the three routing protocols: AODV, DSR, and DYMO over a range of parameters. Some of these parameters were recorded for them being time-varying in nature, while some were a static counter-type parameter.

In conclusion, we believe that DYMO has proved as a better routing protocol. This is substantiated with the many results presented in the previous section where each parameter has been examined and discussed with respect to each protocol.

REFERENCES

[1] *Luc Hogie, Pascal Bouvry, and FrédéricGuinand. 2006. An Overview of MANETs Simulation. Electron. Notes Theor. Comput. Sci. 150, 1 (March 2006), 81-101.*

[2] *Sarkar, Nurul I.; Lol, Wilford G.; 2010. A study of MANET routing protocols: Joint node density, packet length and mobility. Computers and Communications (ISCC), 2010 IEEE Symposium on. 515 – 520.*

[3] *Stuart Kurkowski, Tracy Camp, and Michael Colagrosso. 2005. MANET simulation studies: the incredibles. SIGMOBILE Mob. Comput. Commun. Rev. 9, 4 (October 2005), 50-61.*

[4] *Abdul HadiAbdRahman, Zuriati Ahmad Zukarnain, Performance Comparison of AODV, DSDV and I-DSDV Routing Protocols in Mobile Ad Hoc Networks, European Journal of Scientific ResearchISSN 1450-216X Vol.31 No.4 (2009), pp.566-576.*

[5] *ChristophSommer, Isabel Dietrich, and Falko Dressler. 2010. Simulation of Ad Hoc Routing Protocols using OMNeT++. Mob. Netw. Appl. 15, 6 (December 2010), 786-801.*

[6] *Algorithms and protocols for wireless and mobile ad hoc networks. Hoboken, N.J: Wiley, 2009*

[7] *The handbook of ad hoc wireless networks. Boca Raton: CRC Press, 2003.*

[8] *MiXiM simulator for wireless and mobile networks using OMNeT++. [online]. Available: http://mixim.sourceforge.net/.*

[9] *Varga. OMNeT++ Discrete Event Simulation System. Available: http://www.omnetpp.org/doc/manual/usman.html.*

[10] *Andres Varga and Rudolf Hornig. 2008. An overview of the OMNeT++ simulation environment. In Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems \& workshops (Simutools '08). ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), ICST, Brussels, Belgium, Belgium, , Article 60 , 10 pages.*

[11] *A. Ariza, Implementation of Ad hoc routing protocols for OMNet++, University of Malaga, software available at: http://webpersonal.uma.es/~AARIZAQ/*

[12] *INET Framework Manual.http://inet.omnetpp.org/index.php?n=Main.Manual*

[13] *INETMANET Framework. https://github.com/inetmanet/inetmanet*