A Study of Routing Protocols for Ad-hoc Network

Nitin Kumar\(^1\), Dr. Parvez Ahmed Alvi\(^2\), Dr. Ajay Singh yadav\(^3\), Ms. Anupam Swami\(^4\),

\(^{1,2}\) Banasthali Vidyapeeth, Jaipur, \(^3\)SRM University NCR Campus, Ghaziabad, \\
\(^4\)Govt. Degree College, Sambhal,

Abstract

The work presented here is a summary of the results obtained when routing protocols viz. OLSR, DSDV, and BATMAN 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, radio state and more. We conclude by stating the OLSR emerges as the better protocol of the three examined here.

Keywords— MANET, Routing Protocols, OMNeT++, network simulation, OLSR, DSDV, BATMAN.

1. INTRODUCTION

In our advent into the world of network simulation, we have chosen to take 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.

2. OMNeT++

2.1 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.

2.2 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.
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.

2.3 INET-MANET

INETMANET is based on INET Framework and is continuously developed. Generally it provides the same functionality as the INET Framework, but contains additional protocols 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.

3. 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>
<thead>
<tr>
<th>TABLE I</th>
<th>THE HARDWARE/SOFTWARE SETUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating System</td>
<td>Apple Mac OS X 10.7.3</td>
</tr>
<tr>
<td>Processor</td>
<td>Intel Core 2 Duo – 2.26 GHz</td>
</tr>
<tr>
<td>Memory</td>
<td>5GB</td>
</tr>
<tr>
<td>Compiler</td>
<td>gcc</td>
</tr>
<tr>
<td>Simulation Environment</td>
<td>OMNeT++ 4.2.1</td>
</tr>
<tr>
<td>Simulated using</td>
<td>Cmdenv, Tcl/Tkenv</td>
</tr>
</tbody>
</table>

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.

Simulation Setup

<table>
<thead>
<tr>
<th>TABLE 2</th>
<th>THE SIMULATION SETUP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Playground Dimensions</td>
<td>1000m X 1000m</td>
</tr>
<tr>
<td>No. Of Wireless Hosts</td>
<td>20</td>
</tr>
<tr>
<td>Mobility Model</td>
<td>None</td>
</tr>
<tr>
<td>Max. Channel Power</td>
<td>2.0mW</td>
</tr>
<tr>
<td>Radio Tx Power</td>
<td>2.0mW</td>
</tr>
<tr>
<td>Radio Bitrate</td>
<td>54Mbps</td>
</tr>
<tr>
<td>Broadcast Delay</td>
<td>0s – 0.005s</td>
</tr>
<tr>
<td>Simulation Time</td>
<td>3600s (1 Hour)</td>
</tr>
<tr>
<td>Total Packets Sent</td>
<td>17929.0</td>
</tr>
<tr>
<td>Time Begin</td>
<td>10s</td>
</tr>
<tr>
<td>Message Length</td>
<td>512B</td>
</tr>
<tr>
<td>Message Frequency</td>
<td>0.2s</td>
</tr>
<tr>
<td>Routing Protocols</td>
<td>AODV, DSR, DYMO</td>
</tr>
<tr>
<td>Simulation Style</td>
<td>Cmdenv-express-mode</td>
</tr>
</tbody>
</table>

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 1a and 1b 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.

### 4. RESULTS

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

**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 DSDV, OLSR, and BATMAN on the basis of the contention windows, one must look for a lower value.

![Figure 2](image-url)  
**Figure 2**: The contention window parameter as measured at Host #7, located at an intermediary position between the source and destination hosts.

Figure 2 shows a graph generated by the vector data of the three protocols during simulation in a time-averaged manner. It shows that the three protocols have resulted in a comparable range of contention windows. We have no clear choice on the basis of this single parameter.

**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 3 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 3 shows a minor differential between the three protocols. But, this is one of the scenarios where a longer simulation time has shown its importance. Initially it is seen that either of the three protocols have similar performance, while at about 600 seconds into the simulation OLSR takes a steady lead over BATMAN and DSDV. This phenomenon grows with time and OLSR is continues to have an edge over the rest.
**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 4 and 5 show the time-averaged graphs for destination host and source host respectively. At a glance, the reader is able to deduce that OLSR consumes the lowest power as against BATMAN and DSDV. This remains true for either case: the source and the destination hosts. In fact, the OLSR promises about half the power consumption of BATMAN, while DSDV is marginally higher than OLSR.

![Figure 4: A time averaged measure of Radio State at the destination host.](image)

![Figure 5: A time averaged measure of Radio State at the source host.](image)

**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 6 presents the case for the MAC loss rates for the three protocols over the same network (time-averaged).

A lower MAC loss rate is desirable. A simple understanding of Figure 6 shows BATMAN has the lower MAC loss rate over the observed time period. The reader must note the trend shown by OLSR. It seems to rise with time, creeping towards the curve of DSDV. In terms of absolute values, the MAC loss rates stand between 0.030-0.050 (3-5%). BATMAN has demonstrated a slightly lower MAC loss rate, and while this may go in favor of the BATMAN protocol.

![Figure 6: 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.](image)
Packets received in queue

![Figure 7: A Bar Chart depicting the Packets received by a MAC queue at the source.](image)

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.

Figure 7 depicts the different queue lengths recorded at the source host under the three protocols. BATMAN had the longer queue while OLSR had about half the size of that, while DSDV registering somewhere to the middle of the difference.

Packet Collisions

The idea of a packet collision is self-explanatory. When two or more packets cross each other’s 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 8 shows statistics related to the packet collisions in the network on the three protocols. Once again BATMAN performs poorly by having the most number of packet collisions.

![Figure 8: Packet Collisions recorded.](image)

Packets Sent without Retry

In an ideal scenario, a packet should be sent only once. When this is not so, a packet will be sent again. This would be counted as packets sent with retry. We are interested in the number of packets a protocol was deliver on the first occasion, without a retry. This is achieved on different levels by the three protocols examined.

Figure 9, illustrates that data recorded at an intermediary host #13. According to the figure, OLSR has the higher number of packets sent without a retry. While BATMAN and DSDV have lower figures for this parameter. OLSR has the advantage here.

![Figure 9: Packets sent without a retry – Host #13.](image)

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.
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