

CHAPTER 4

**IMPLEMENTATION, SIMULATION RESULTS AND
ANALYSIS**

4.1 Introduction

The performance metrics of AODV and AODV-LP (AODV with Link Prediction) are compared using simulation. This chapter describes the simulator used for this work (NS2) and our implementation, followed by the configuration of simulation parameters, performance measurements, and an explanation of results and analysis.

4.2 Network simulator version 2 (NS2)

Network Simulator (Version 2), widely known as NS2, is simply an event-driven simulation tool that has proved useful in studying the dynamic nature of communication networks. Simulation of wired as well as wireless network functions and protocols (e.g., routing algorithms, TCP, UDP) can be done using NS2. In general, NS2 provides users with a way of specifying such network protocols and simulating their corresponding behaviors. NS2 consists of two key languages: C++ and Object-oriented Tool Command Language (OTcl). While the C++ defines the internal mechanism (i.e., a backend) of the simulation, the OTcl sets up simulation by assembling and configuring the objects as well as scheduling discrete events (i.e., a frontend). The C++ and the OTcl are linked together using TclCL.

Figure 4.1 shows the basic architecture of NS2. NS2 provides users with an executable command “ns” which takes one input argument, the name of a Tcl simulation scripting file. In most cases, a simulation trace file is created and is used to plot graph and/or to create animation [48].

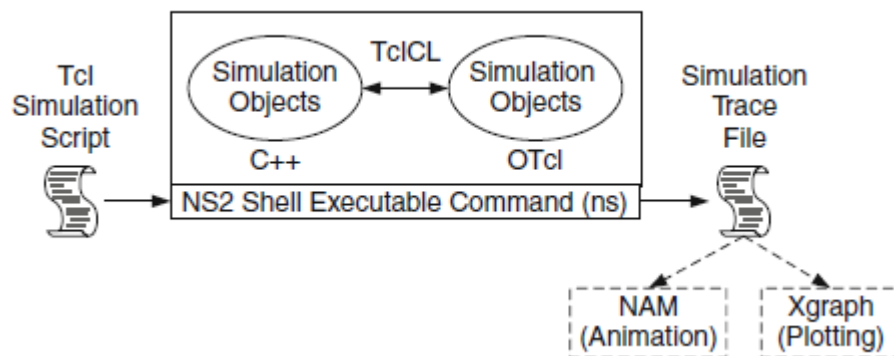


Figure 4.1 Basic architecture of NS

4.2.1 Network components

The basic classes used to define the architecture and topology of network are [47]:

- 1) Node

A Node class is an Otcl class, this class and its methods are defined in `tcl/lib/ns-node.tcl` file, a node is a collection of classifiers and agents.

2) Link

A link is used for connect the nodes, it models the transmission system, the link is mainly characterized by a propagation delay and bandwidth. It is an Otcl class which includes a set of components derived from the Connector class. This class and its methods are defined in `tcl/lib/ns-link.tcl` file.

3) Agent

An agent is another component of node, it models the constructors and consumers of IP packets. The agent class provides methods useful for the development of the transport layer. This class is both in the interpreter and simulator. This is the base class for defining new protocols in NS. It provides local and destination address, functions to generate packets and interface to the Application class.

4) Packet and Header

Packet management in NS implement three classes:

- Packet class models the data units exchanged between network components.
- PacketHeaderClass is derived from TclClass and provides methods to access and communicate with the interpreter. It serves to locate in the packet header of a particular protocol.
- PacketHeaderManager class is used to manage packet headers that are available in the simulator.

4.2.2 Simulation elements

There are three main simulation elements [47]:

➤ Simulator

The simulation is configured, controlled and operated using the interface provided by Otcl Simulator class. A simulation script always starts by creating an instance of this class with command `set ns_ [new Simulator]`.

➤ Event scheduler

The scheduler is defined in `scheduler.{h,cc}` file, The scheduler selects the closest event in terms of time, executing treatments, advancing time simulated and move forward to the next event etc.

➤ Timer

There are two mechanisms of timers in NS:

- ✓ One for the scheduling of events generated by the interpreter. It is defined in `tcl/ex/timer.tcl` file.
- ✓ A second set in C++ for the simulator components.

4.2.3 Simulation Outputs

➤ Trace

There are two types of traces: traces made from a queue of a link and trace of variables. The code of trace functions is respectively in `trace.cc` and `tclcl/tracedvar.cc`. Trace file is a text file Patterned Structured in lines [47].

➤ Nam

After simulation, NS2 outputs either text-based simulation results. To interpret these results graphically and interactively, tools such as NAM (Network AniMator) and XGraph are used. To analyze a particular behavior of the network [48].

4.3 Implementation

Our system includes two parts and is implemented using two languages

4.3.1 C++ implementation

Main C++ Files for AODV Stored in the Directory `~ns/aodv/`

<code>aodv.cc,h</code>	Main definition of AODV routing agents
<code>aodv_packet.h</code>	Packet header of AODV routing agents
<code>aodv_rtable.cc,h</code>	AODV route entry and routing table
<code>aodv_rqueue.cc,h</code>	Buffer which stores data packets during a route discovery process

Our proposed algorithm AODV-LP (AODV with Link Prediction) is implemented by modifying three files of the original AODV:

1. `aodv.h` file

A header file, it contains and defines timers, signature functions (prototypes), and AODV routing agent which is responsible for creating, sending, receiving, processing, and destroying routing packets.

2. `aodv_packet.h`

Packet header file declares different packet types and its fields used by AODV protocol. We have added two fields: `MyRSSI` and `GetTime` to HELLO packet format, and `NodeWllBk` to REQUEST packet format.

3. aodv.cc file

This is the body class of protocol, it implements the behavior of AODV protocol, including timers, Tcl Hooks, and all functionality of routing agent with its methods. Most of our work has been added in this file. We have, first, implemented some functions which define the behavior of AODV-LP and then we have modified some exiting functions such as: `recvHello(Packet *p)`, `recvRequest(Packet *p)` and `sendRequest(nsaddr_t dst)`.

4.3.2 Tcl implementation

We have simulated the standard AODV routing algorithm and (AODV-LP) link prediction using many TCL scripts in order to get optimal results, this is done by extracting the average of results from trace files. Figure 4.2 shows the evaluation process of the two protocols.

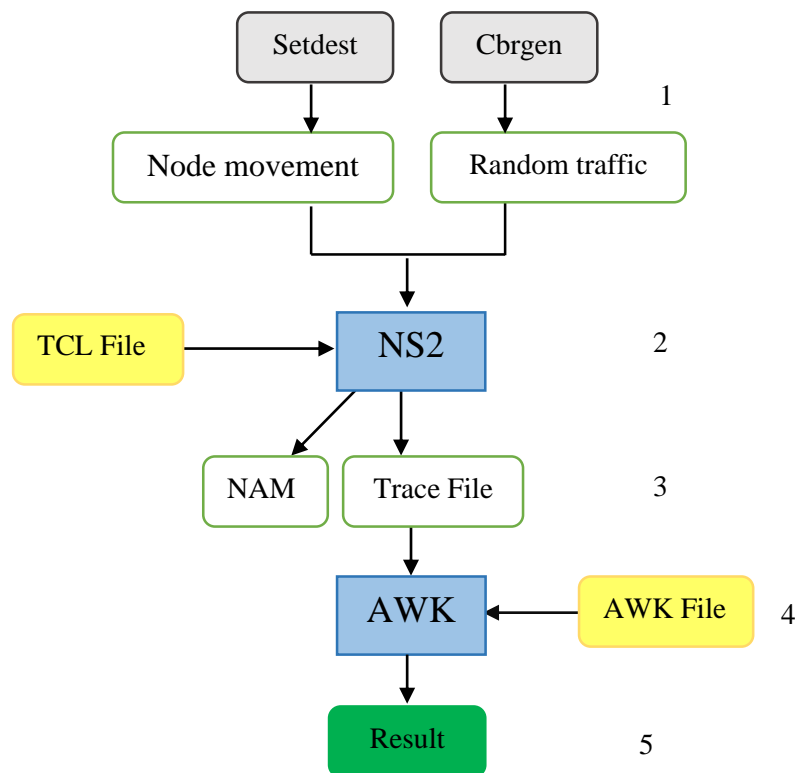


Figure 4.2 Simulation process

The scheme above is based on 5 operations as explained below:

1. Setdest and cbrgen scripts

➤ Setdest

A node movement generator script which generates movements of nodes automatically in a separate file, setdest is also used to:

- Generate initial nodes positions.
- Move nodes from one position to another using setdest (Set destination) with a specific speed

The binary of this tool is located under the following directory ~ns/indep-utils/cmu-scen-gen/.

It is used as follows:

```
./setdest -n <num_of_nodes> -p <pausetime> -s <maxspeed> -t <simtime>  
-x <maxx> -y <maxy> > <outdir>/<movement-file>
```

Example

```
./setdest -n 50 -p 5.0 -s 20.0 -t 100 -x 1000 -y 1000 > user_setdest.tcl
```

The file begins with the initial position of the nodes and goes on to define the node movements

```
$ns_ at 2.000000000000 "$node_(0) setdest 90.441179033457 44.896095544010  
1.373556960010"
```

➤ Cbrgen

Generates Random traffic connections in a separate file. It allows to:

- Create connections between the nodes, one can specify the maximum number of connections to be created for all the nodes in the network
- Create the type of agents between the nodes (cbr or tcp)
- Set the rate at which the packets are transmitted.

The command line looks like the following:

```
ns cbrgen.tcl [-type cbr|tcp] [-nn nodes] [-seed seed] [-mc connections]  
[-rate rate]
```

Example

```
ns cbrgen.tcl -type cbr -nn 10 -seed 1.0 -mc 8 -rate 4.0 > cbr-10-  
test.tcl
```

From cbr-10-test file (into which the output of the generator is redirected) thus created, one of the cbr connections looks like the following:

```
# 2 connecting to 3 at time 82.557023746220864
#
set udp_(0) [new Agent/UDP]
$ns_ attach-agent $node_(2) $udp_(0)
set null_(0) [new Agent/Null]
$ns_ attach-agent $node_(3) $null_(0)
set cbr_(0) [new Application/Traffic/CBR]
$cbr_(0) set packetSize_ 512
$cbr_(0) set interval_ 0.25
$cbr_(0) set random_ 1
$cbr_(0) set maxpkts_ 10000
$cbr_(0) attach-agent $udp_(0)
$ns_ connect $udp_(0) $null_(0)
$ns_ at 82.557023746220864 "$cbr_(0) start"
```

2. Creating Tcl file and simulation

After generating the network topology (by setdest file) and network traffic (by cbrgen file), this two files are added into main Tcl file which also contains the initialization of network simulator, output files creation, scheduling events...etc.

3. Simulation outputs

When the execution is terminated, a trace file is generated which contains the events trace, and also a NAM file is generated.

4. Trace file Analysis

NS2 doesn't provide a mechanism for analyzing the outputs, we have used AWK script to analyze the output from the generated trace file towards the end of simulation.

5. Results of analysis

The result represents the network performance in terms of every metric used in simulation. This result represents final step of our simulation.

4.4 Simulation parameters

The following configuration parameters are set to run the experiments. These parameters are already available in the simulator NS. The network simulator NS implements the four layers of TCP/IP model which are physical, data link, network (routing), and application.

Table 4.1 summarizes the simulation parameters used in our simulation.

Parameter	Value
Routing Protocol	AODV
Channel Type	Wireless Channel
Radio-Propagation Model	Two Ray Ground
Network Interface Type	Wirelessphy
Mac Protocol	802.11
Interface Queue Type	Droptail/Priqueue
Link Layer Type	LL
Antenna Model	Omniantenna
Mobility Model	Random Waypoint
Simulation Area	1000 x 1000 m ²
Total Nodes	50, 100, and 150
Max Velocity	5, 10, 15, 20 m/s
Pause Time	0, 5, 10, 25, 50, 100 s
Traffic Type	Constant Bit Rate (CBR)
Transmission Rate	1, 2, 10, 20 Packets/s
Total Connections	1 Connection
Transmission Range	250 m
Packet Size	512 Bytes
Simulation Time	100 s

Table 4.1 Simulation Settings

4.5 Performance measurements

The performance of AODV-LP (AODV with Link Prediction) is compared with original AODV protocol and evaluated in terms of packet delivery ratio and average end-to-end delay as a function of number of nodes, node mobility (mobility speed), pause time and transmission rate. The number of nodes was varied from 50 to 150, node velocity from 5 to 20 m/s, pause time from 0 to 100s and transmission rate from 1 to 20 Packets/s. At a time, one

variable was changed and other three were kept constant. The three parameters which are kept fixed, are assumed to take values as follows, network size = 50 nodes, node velocity = 20 m/s, pause time = 5 s and transmission rate = 10 Packets/s.

Packet delivery ratio: the ratio of the number of delivered data packets to the destination. This illustrates the level of delivered data to the destination. The greater value of packet delivery ratio means the better performance of the protocol.

$$\sum \text{Number of packet receive} / \sum \text{Number of packet send}$$

End-to-end Delay: the average time taken by a data packet to arrive to the destination, it includes all possible delays caused by buffering during route discovery, queuing at interface queue, retransmission delays at MAC layer, propagation and transfer time.

$$\sum (\text{arrive time} - \text{send time}) / \sum \text{Number of connections}$$

4.6 Results and discussion

This section describes the simulation results, and provides a performance comparison between AODV-LP and standard AODV in terms of End-to-End Delay and Packet Delivery Ratio metrics, with discussion and analysis of each result.

4.6.1 Impact of Transmission Rate

4.6.1.1 Average End-To-End Delay Performance Metric

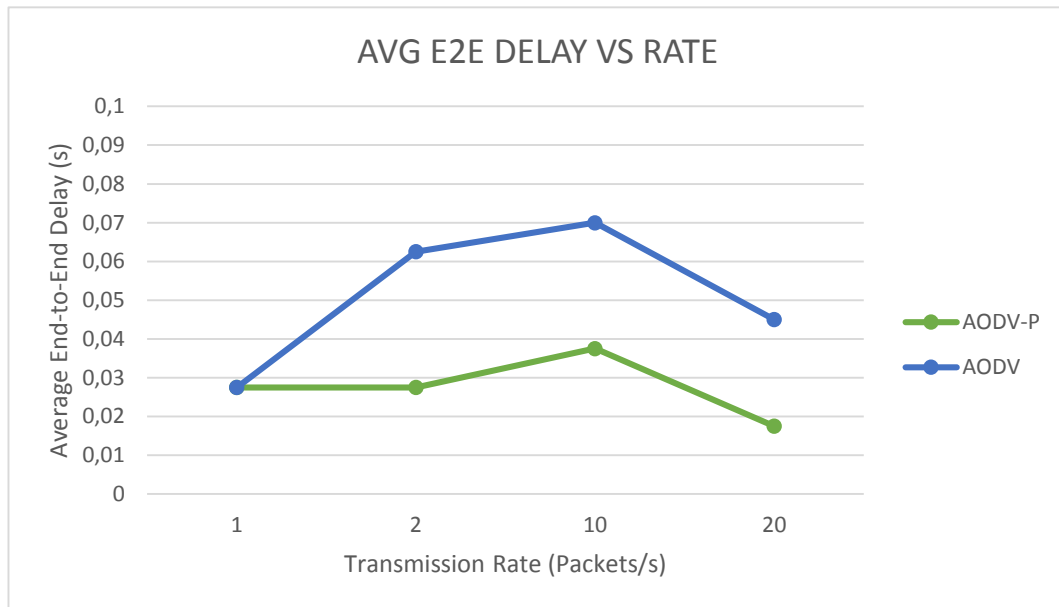


Figure 4.3 Average End-to-End Delay versus Transmission Rate

From Figure 4.3, we observe that there is a decrease in end-to-end delay in AODV-LP as compared to AODV because alternative routes are discovered before the route failures.

However, end-to-end delay increases as the transmission rate increases in AODV-LP and AODV because more transmission of packets causes more collisions.

Consequently, the increase of transmission rate causes increase bandwidth consumption, this causes more collisions, thus retransmission of packets which increase overheads and end-to-end delay.

4.6.1.2 Packet Delivery Ratio Performance Metric

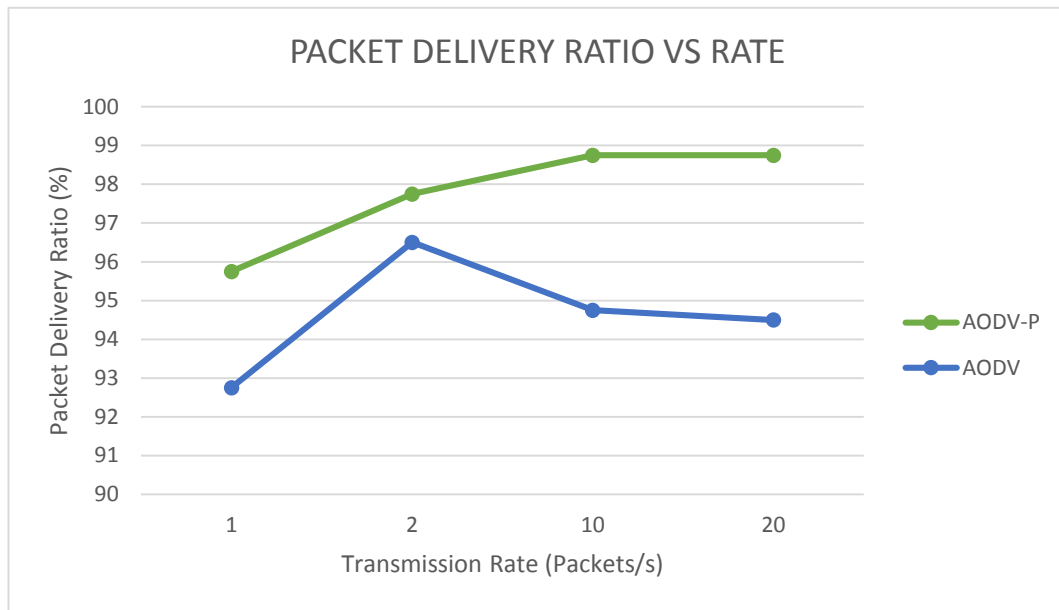


Figure 4.4 Packet Delivery Ratio versus Transmission Rate

Figure 4.4 shows variation of packet delivery ratio with increasing transmission rate. Results show that packet delivery ratio is better in AODV-LP as compared to AODV. It happens because in AODV-LP, alternative routes are discovered before the route failures, and more data is successfully delivered to the destination.

The result also shows that the packet delivery ratio decreases in AODV as the transmission rate increases from 2 to 20 Packets/s, and in AODV-LP the packet delivery ratio increases as the transmission rate from 2 to 10 Packets/s and it stays constant from 10 to 20 Packets/s.

Consequently, more transmission rate causes more collision and more retransmission packets, furthermore it causes more overhead and when link failures occur the overhead increases for another route discovery, this increases collision which increases packets drops thus low packet delivery ratio.

4.6.2 Impact of Mobility Speed

4.6.2.1 Average End-To-End Delay Performance Metric

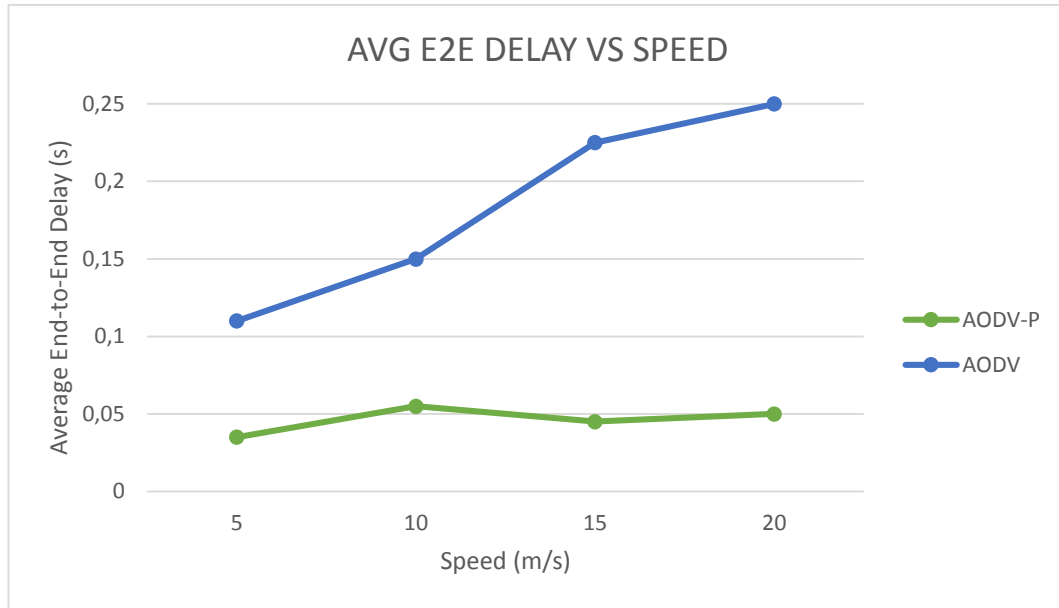


Figure 4.5 Average End-to-End Delay versus Node Speed

In Figure 4.5, we observe that the end-to-end delay increases when node velocity increases for AODV as from 5 to 20 m/s, and for AODV-LP there is a small increasing of end-to-end delay when node velocity increases only from 5 to 10 m/s, after that, almost constant until 20 m/s.

Increasing of end-to-end delay in terms of increasing the speed indicates that more route failures occur for fast moving nodes, so AODV-LP not affected by the increase of mobility speed, because it predicts the link before it breaks.

Therefore, overheads of new route discovery lead to increase of end-to-end delay.

4.6.2.2 Packet Delivery Ratio Performance Metric

Figure 4.6 shows variation of packet delivery ratio with increasing node velocity. Results show that packet delivery ratio is better in AODV-LP as compared to AODV. It happens because in AODV-LP, alternative routes are discovered before the route failures, and more data is successfully delivered to the destination.

The result also shows that the packet delivery ratio decreases as the node velocity increases because faster mobility of nodes causes more route failures.

Consequently, more route failures result in packet drops and thus low packet delivery ratio.

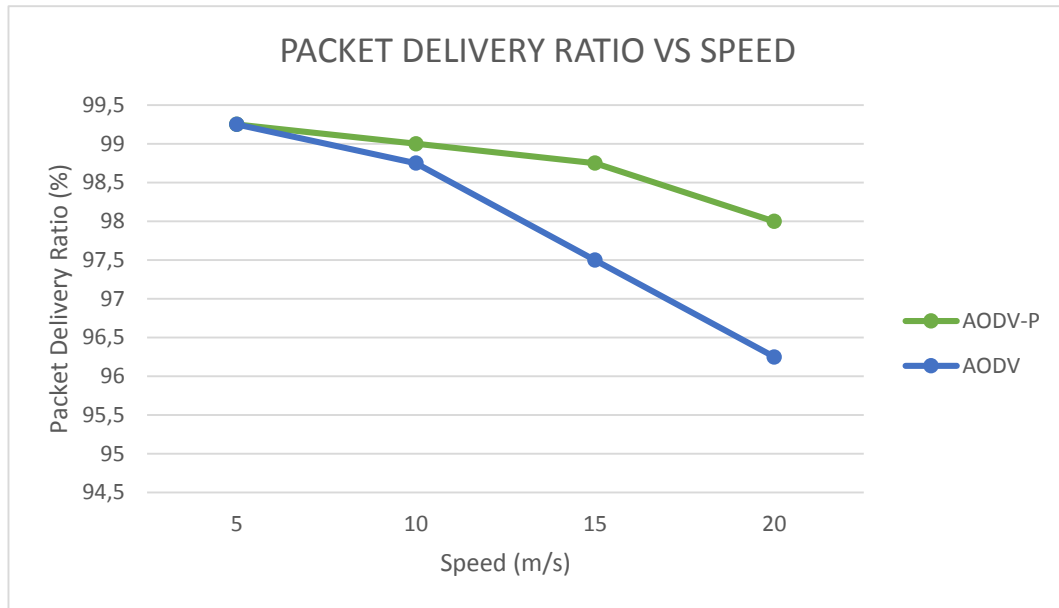


Figure 4.6 Packet Delivery Ratio versus Node Speed

4.6.3 Impact of Pause Time

4.6.3.1 Average End-To-End Delay Performance Metric

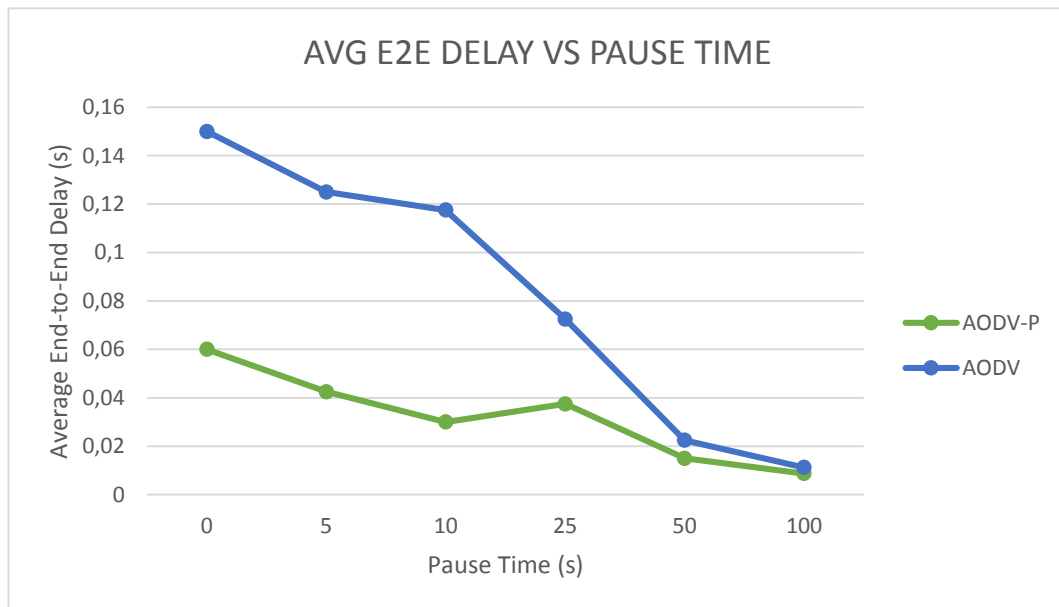


Figure 4.7 Average End-to-End Delay versus Pause Time

Figure 4.7 shows decrease in end-to-end delay in AODV-LP as compared to AODV due to advance route discovery in case of route failures. However, end-to-end delay decreases with increase in pause time in AODV-LP and AODV, because the increase of pause time is the decrease of node mobility from 0 to 50 s, until 100 s where end-to-end delay is the same for AODV and AODV-LP because in this case, network will be static.

Therefore, more node mobility causes more route failures which increases overheads of new route discovery, and local repair this causes more decrease of end-to-end delay.

4.6.3.2 Packet Delivery Ratio Performance Metric

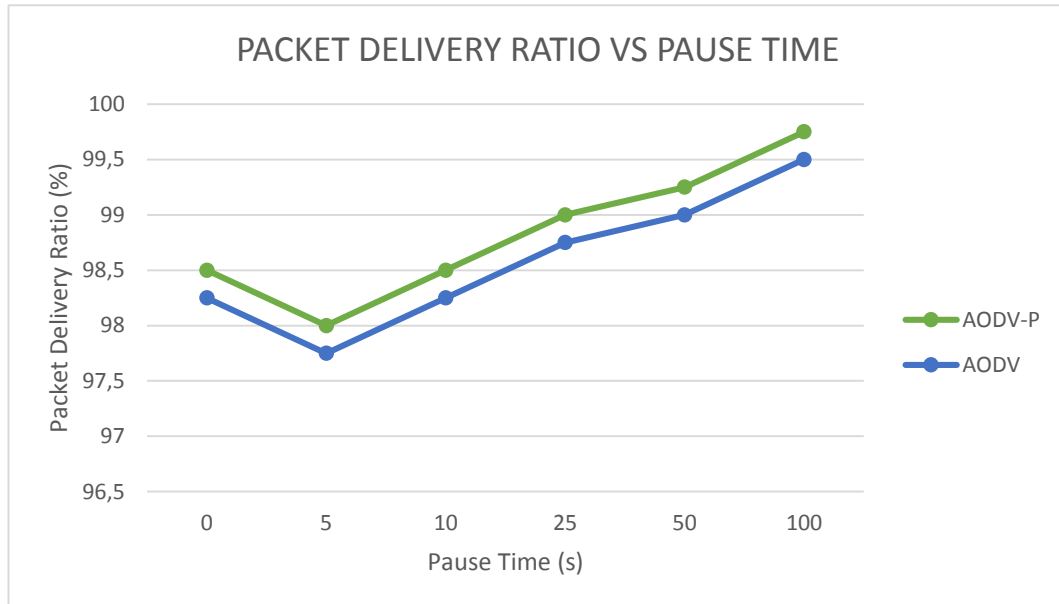


Figure 4.8 Packet Delivery Ratio versus Pause Time

Figure 4.8 shows variation of packet delivery ratio with increasing pause time. Results show that packet delivery ratio is better in AODV-LP as compared to AODV, because in AODV-LP, alternative routes are discovered before the route breaks, and more data is successfully delivered to the destination, until the network will be static, packet delivery ratio will be in higher value.

The result also shows that the packet delivery ratio increases as the pause time increases because less movement of nodes decrease route failures. Consequently, less mobility result in less packet drops and thus high packet delivery ratio.

4.6.4 Impact of Number of Nodes

4.6.4.1 Average End-To-End Delay Performance Metric

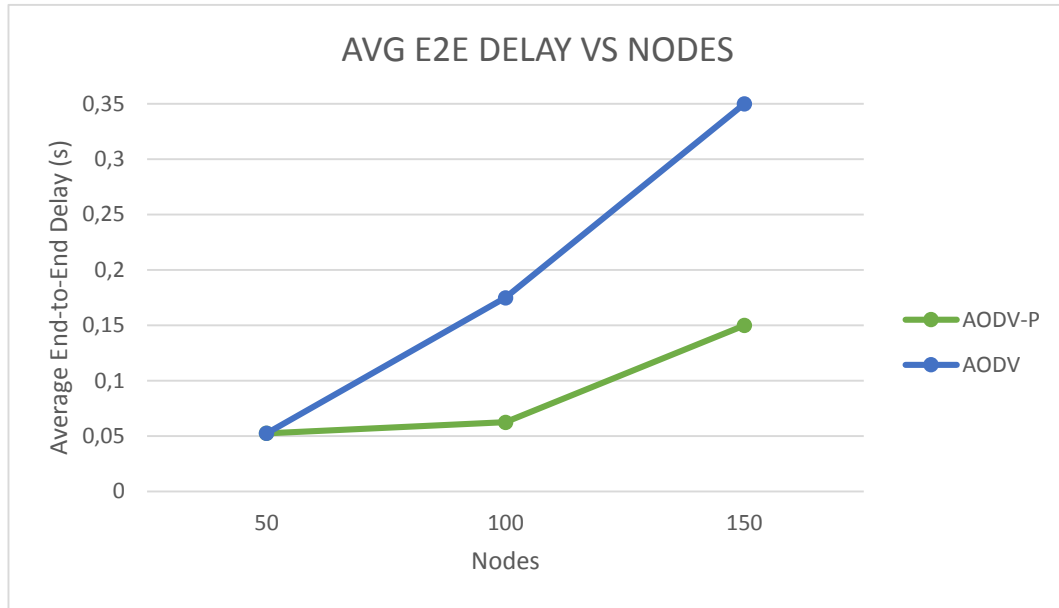


Figure 4.9 Average End-to-End Delay versus Number of Nodes

The end-to-end delay is an average of difference between the time when data packet is generated and the time when data packet is received at its destination. Figure 4.9 shows decrease in end-to-end delay in AODV-LP as compared to AODV due to advance route discovery in case of route failures. However, end-to-end delay increases with increase in the network size in AODV-LP and AODV because high node density increases collisions, which results in retransmission of packets.

Consequently, high node density causes more overheads (routing information), thus reducing bandwidth of network, this decrease reduces the data transmitted over the network and increases collisions which results retransmission of packets thus increase of end-to-end delay.

4.6.4.2 Packet delivery ratio performance metric

Figure 4.10 shows variation of packet delivery ratio with increasing network size. Results show that packet delivery ratio is much the same for both AODV-LP and AODV when number of nodes is 50, then a little bit better in AODV-LP as compared to AODV when network size is 100 nodes, after that, Not much higher for AODV than AODV-LP. It happens because in AODV-LP, alternative routes are discovered before the route failures, and more data is successfully delivered to the destination.

However, AODV-LP and AODV give smaller delivery ratio as network size increases, since it has more route failures which results in packet drops. Increased node density causes more contentions and collisions due to more neighboring nodes. Therefore, Due to more node density, the delivery ratio decreases by increase of packet drops.

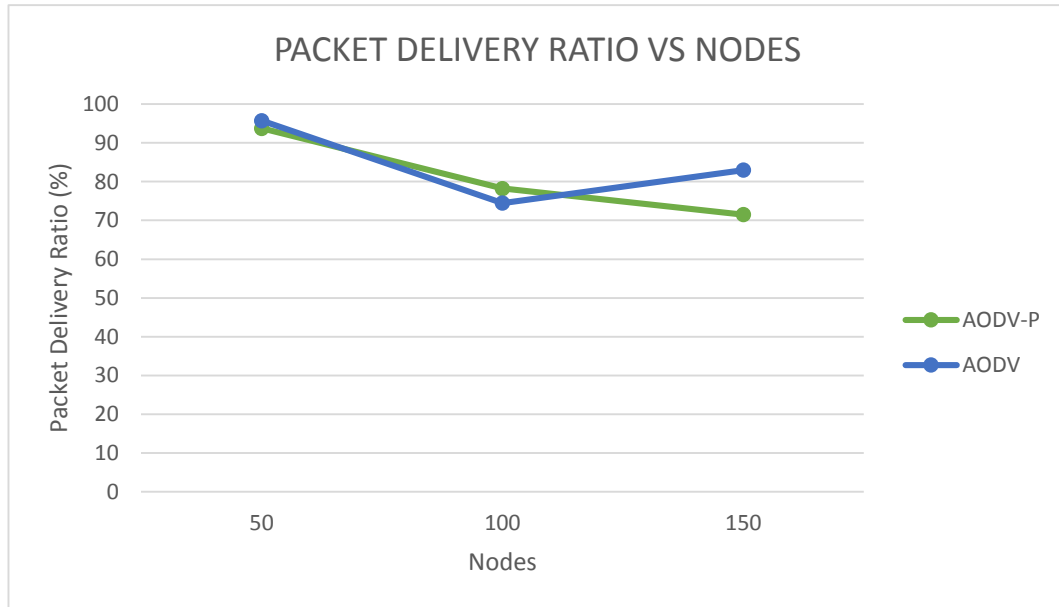


Figure 4.10 Packet Delivery Ratio versus Number of Nodes

4.7 Conclusion

In this chapter we have described the implementation of our protocol AODV-LP (AODV with Link Prediction) and then presented the simulation results. Our implementation includes two parts using two languages (C++ and TCL) under NS2 simulator (Network Simulator 2). We have configured initial parameters of our protocol, and defined the performance measurements. The performance of the proposed AODV-LP has been evaluated and compared with AODV. The simulation results showed that the proposed algorithm outperforms the original AODV protocol and in terms of end-to-end delay and packet delivery ratio.