Table of Contents

List of tables

List of Figures

Chapter 1 : Introduction

1.1. Overview

Nowadays Wireless sensor Networks technology has become tremendously vital in a diversity of aspects, since, it has become intensively necessary in many different domains where the use of this cutting edge technology turned out to become not only an option but crusial, whether in ordinary utilization, in health environment or in Military use. Indeed, there is numerous diverse sorts of Wireless sensor Technology, that differs regarding the domain of use, such as Zigbee Wireless Sensor Networks who are better for energy comsumption efficiency, or Wi-Fi Wireless Sensor Networks who offers higher data rates and enhanced performance, all of them have its special domain of application.

Apart from this introductory chapter, this dissertation will contain three more chapters. Initially the literature review chapter which is going to underline and stress the diverse significant theoretical and background points that will be required to accomplish this M.Sc.A. final project, and which have been utilized in this research project, this chapter will give a theoretical point of view of the need behind the Wireless Sensor Network as well as some theory behind diverse WSN technologies, especially the routing protocols.

Afterward, in the findings and Discussion chapter the main relative findings of the research will be discussed; then, from that discussion some conclusion might be

made such as whether or not the number of nodes utilised in the WSN affects the Network performance, And what routing protocol is performing better and so on and so forth, also, this chapter will enclose some figures and tables which will be presented to show the result of the testing and simlation steps. And finally, in the conclusion chapter, a summarise of what was performed throughout this report will be presented; then, some recommendations which might improve the performance of the WSN, will be given, so that, it probably will help to achieve further research in the domain of WSN.

1.2. Project aims and objectives

The aim of this M.Sc.A. Project is to design and simulate different WSN topologies for the transmission of video data in the context of intelligent streetlights. Routing methods will be compared to achieve the required throughput and to meet specific deployment constraints. Network simulations will be carried out to validate the theoretical study.

1.3. Research Approach

For this final project, the research approach is basically a primary based research whilst it will include some second based research aspects as we are going to design, build and test a different Wireless sensor Network topologies.

In addition, scientific research is defined as : studying a subject in detail for discovering new information or to find out a new interpretation [12]. Moreover, for this final project both quantitative and qualitative approaches will be utilized and basically, the project will be found on a primary research aspect.

Chapter 2 : Literature Review

Like many other technologies Wireless Sensor Networks is the fruit of continuous effort and research lead by military research institutions, for instance, in 1978 during the distributed Sensor Nets workshop arranged by DARPA [1], The aim of this workshop was to foster research and innovation in the domains related to sensor networks such as distributed programs and signal processing [1].

Then in 2001, the notion of Wireless Integrated Network Sensors (WINS) was introduced by the university of California with cooperation of Rockwell Science center [1].

The smart sensing system is composed by a digital signal processing circuit, interface circuits wireless radio,microcontroller, integrated multiple sensors and a CMOS chip [1].

In addition a WSN Project called the smart dust project at the university of California at Berkeley aimed to invent a very tiny sensor nodes termed mote [1]. The purpose of this research was to show that very small devices such as the size of a dust particle or a grain of sand are fully capable to include a thorough sensor system.

In 2000 Rabaey et al. worked on the PicoRadio project which aimed to design a low-power wireless sensor instruments, that consume energy in a significantly very low rate and which are able to get energy autonomously from the functioning environment like solar energy [1].

And in 2005 Calhoun et al concentrated on the MIT μAMPS project which highlighted low-power for sensor nodes in two aspects, material and program. It involved the utilization of dedicated microcontrollers for data processing methods and for active voltage scaling in order to decrease power consumption in terms of the software side [1].

Even though most of the research work has been done by academic organisations and universities, throughout recent years, numerous startups and firms utilized on WSN have emerged and contributed on fostering this research field. For instance Ember Corporation , Dust Networks, Sensoria are some of these companies [1].

These firms offers the option to buy WSN instruments that fits the different real life needs with various programming and data management options [1].

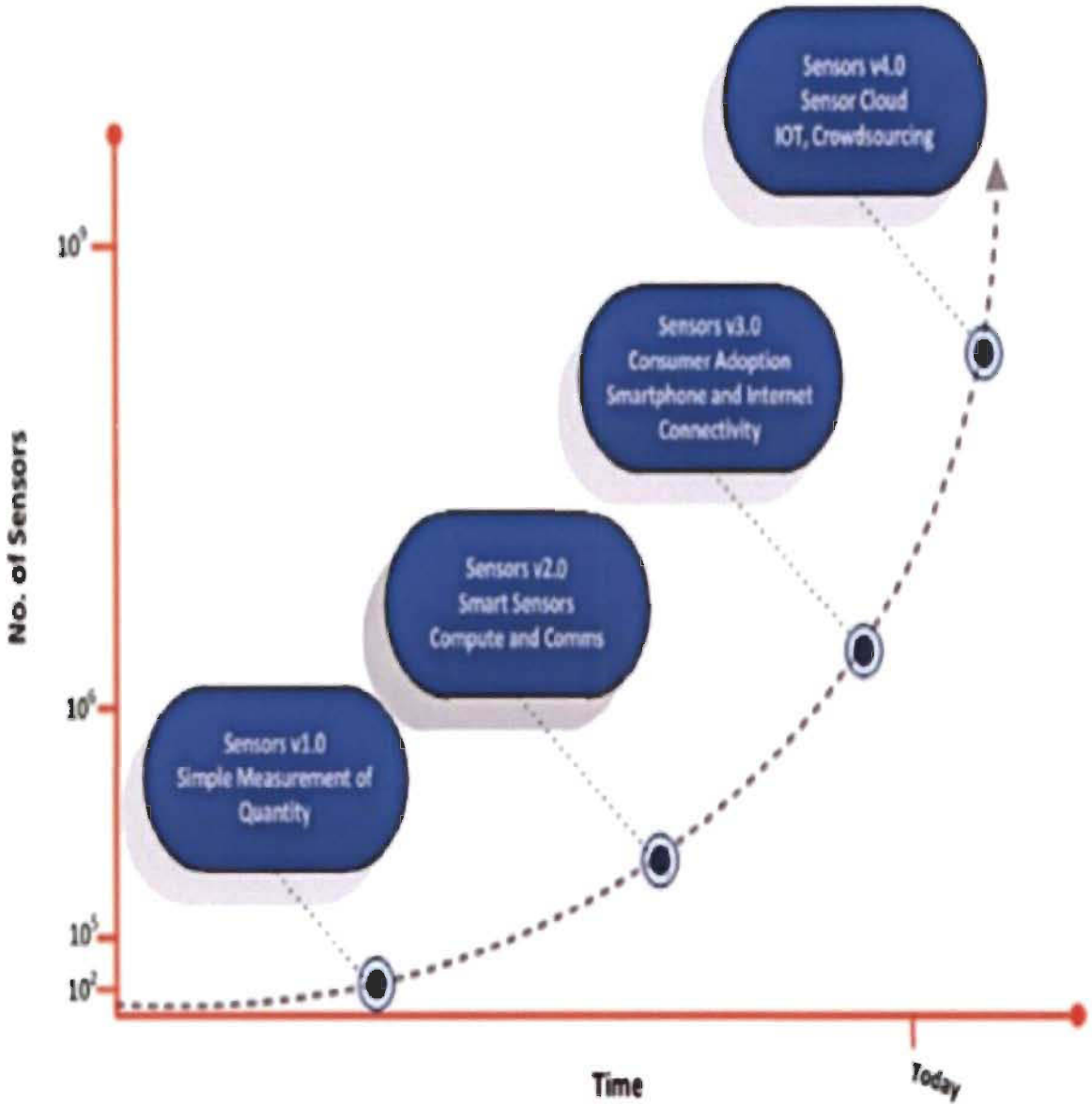


Figure 1 Evolution of sensor during last decades [4]

2.1. WSN Applications

WSN have different kind of applications in several domains such are health, Military, national security and the Internet of things

2.1.1 Health

Health Problems related to modern life style have risen during the last decades and the need to get a technological solution has also increased. One of these solutions is WSN [1].

A research on heart diseases made at Framingham institute came to the conclusion that many cardiovascular diseases can be avoided due to physical activities [1]. And a new publication proved that around 10 percent of the total number of deaths in 2008 are caused by lack of activity [1].

Recent research acclaims 150 minutes of sport every week for adults. But about 33 % of adults do not practice enough training and sport, thus these people are significantly more vulnerable to diabetes and cardiovascular diseases [1].

Eating habits have altered throughout the last fifty years. Also the popularity of fast and processed foods increased steadily, those kinds of foods contains more amount of fat salt and sugar.

Meals based on meat has become very common, and the consumption of vegtables and fruits declined, thus, the amount of calories rose triggering increase obesity and diseases related to it.

All these health problems mentioned above, have to be tackled by research and technologies, where cams the need to WSN in this filed. For instance wireless sensors can be connected with the patient's organs and send information to the doctors in real time.

2.1.2 National Security

Nowadays national security risk of non-classical crimes, is considered as a steady issue for both citizens and governments, These dangers vary from chemical and biological to even nuclear and radiological.

Unceasing observing and extra caution is consequently needed to thwart these kind of threats.

Even though detection methods which relies on laboratory analysis shows outstanding sensitivity and results, these laboratory are usually far away from the menace zone, therefore, this many cause many issues such as substantial delays and struggle in terms of information gathering.

Hence the fundamental need of wireless sensors deployed in the threat zone and able to gather information promptly and send it in real time to the laboratory or the research institutions.

In this case WSN are required to identify risks in different places and conditions, such as on land or in air and so on.

2.1.3 The Internet of Things

The Internet of things has bounded our lives in a variety of ways. Thanks to ubiquitous internet connectivity, and sophistication in Information technologies, IoT has become a de facto need for many individuals, institutions and companies.

This fact will change drastically the life style of many human being in many aspects like distraction, employment and education.

WSNs contribute significantly in interacting with the outside world,

Like many other technologies IoT is not an exception in the sense that its success depends on binging perceptible advances and enhancements to human being.

It is expected that WSN are going to be vital in delivering data flows, that IoT's projects will be developed.

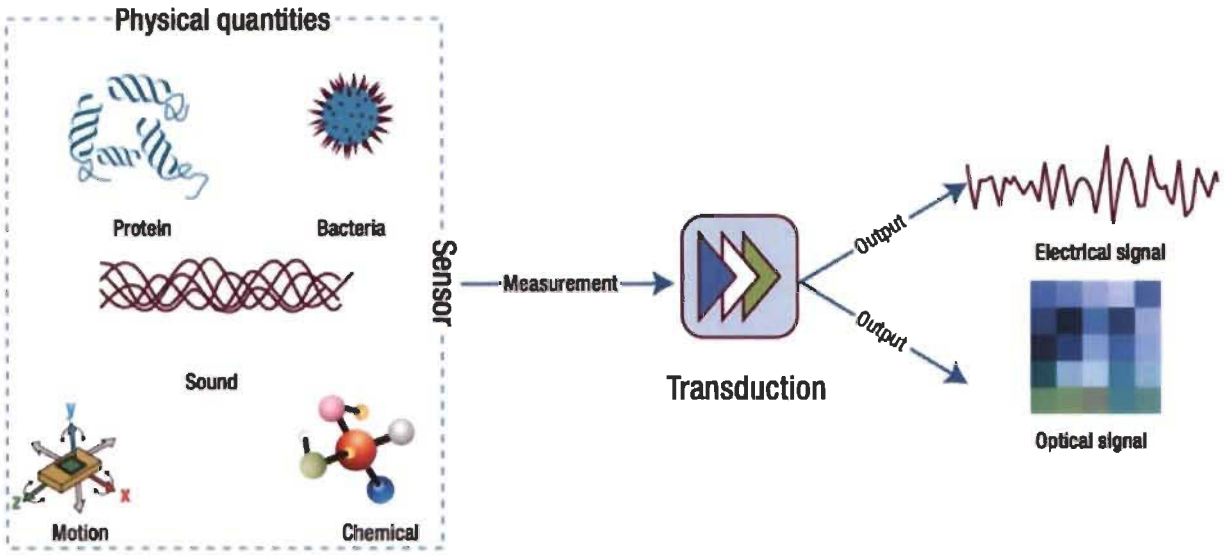


Figure 2 An Exemple of a WSN Application in IOT Internet of Things [1]

2.2. Rooting Protocols

2.2.1 DSR

In 1994 Johnson introduced the DSR, DSR stands for Dynamic source routing, and it uses what is called "route discovery" and "route maintenance" technics.

In DSR every node preserves a path cache with new information that changes constantly when node acquires new paths.

A node willing to direct a packet will primarily examine its route cache to find out if it previously had a route to the endpoint. And this is alike the AODV protocol.

And in the case where no available route in the cache, the sender starts a path

discovery procedure through a path request packet, which encloses the destination's address, the source's address and a sole application ID.

When the request spreads across the network, every node enclosures its particular address inside the application packet prior re-spreading it.

As a result, a request packet registers a path containing the entire nodes it has call on. And once a node gets a request packet and discovers its sole address logged in the packet, it throw-outs this packet and stop spreading it more.

A node preserves a cache of freshly forwarded application packets; saving their sender request unique IDs and addresses, and rejects every identical request packets.

When a request packet reaches its endpoint, it will save the whole route from the source to the endpoint.

In what so-called "symmetric network", the target node is able to unicast an answer packet, covering the gathered path information, back to the source node using the rigorous identical route as used by the request packet.

And in the case of what is known as "asymmetric network, the destination is able to originate a path discovery method from the source to the endpoint.

As soon as the reply packet reaches the source, the source node will be able to save the novel path inside its cache and begin conveying packets to the endpoint.

The DSR routing protocol, is also alike the AODV in another point, they both retains a path conservation method founded on error messages, that are produced when the link layer senses a communication breakdown because of an interrupted link.

Each packet in DSR conveys path information, unlike AODV. This advantage of DSR over AODV permits to in-between nodes (Between source and destination) to insert novel paths preemptively to their personal caches.

2.2.2 AODV

Ad hoc on- Demand Distance Vector (AOVC) Protocol was invinted by Pekins And Royer in 1999.

Nodes do not preserve routing details and do not contribute in cyclic routing table updates.

AODV utilizes what so called broadcast route finding method. When the source lacks routing details in its table, the AODV's route finding procedure, is started when a source node wants to send data to alternative node.

In order to do this, the source node transmits a route request (RREQ) packet to other nodes that are near it, these nodes encloses the addresses of the endpoint and the source, the hop total value, a transmission ID and dual sequence numbers.

The transmission ID will be increased by one when the source node discharges a novel RREQ packet that is joined to the address of the source node in order to solely identify a route request.

As soon as a node obtains a route request packet, a node, which owns an existing path to the endpoint node, replies by transferring a route reply (RREP) message straight to the node that was behind the launching of the route request (RREQ).

Then, the route request is resent by the node to its in-between neighbors. Thus an identical route request (RREQ) is rejected.

Every node in the topology preserves its private sequence number. And If a source node delivering a route request packet it already have its specific sequence number.

Therefore, In-between nodes do not respond to a RREQ unless the sequence number of their path to the endpoint node is (>=) greater than or equal to the endpoint order number stated in the route request packet.

Whenever a route request is resent, the in-between node saves the neighbor's node location, from which the route request was obtained, thus creating an opposite route from the source node to the endpoint node.

When the Route Reply return to the source node, every in-between nodes establish a forward pointer toward the node where the Route reply was received and registers the lates which the RREP

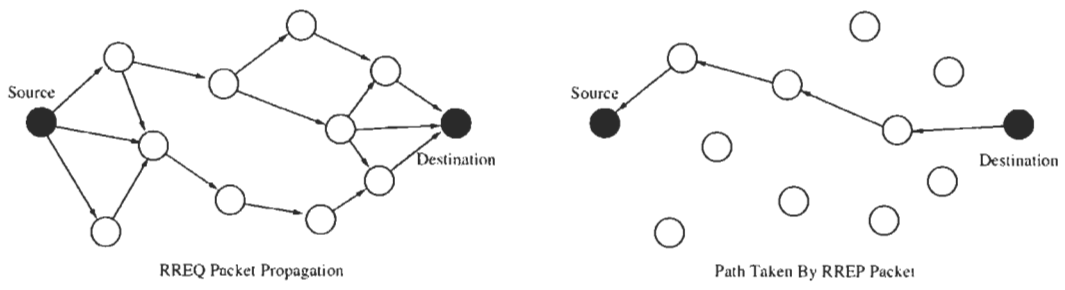


Figure 3 Path discovery process of the AODV routing protocol [1].

This figure above illustrates the procedure of path discovery, it gets the source

nodes after multiple hops [1].

The lifecycle of unutilized routes is reduced using a timer embedded in every node's routing table.

A "Hello Message" is regularly swapped to show how their links are performing.

Throughout the process, whenever a link interrupts, the midway node near to the source, detects the problem and sends to the source what is called a route error packet.

As soon as the source obtains a route error packet, it refreshes the path discovery procedure.

With the AODV routing protocol, routes are solely created when required, which helps significantly to optimize route table and to prevent unnecessary updates and interactions with unused routes.

Whilst, AODV needs regularly to swap the "Hello messages" with other nodes that are close to it.

When a source node wants to send data, not always it has a proper route in its routing table. Therefore, a delay is caused because of that.

In the case of the reverse path, the route created between the source and destination is the opposite route, thus AODV routinhg protocol considers that all paths are symmetric.

2.2.3 DSDV

Destination-Sequenced Distance-Vector Routing (DSDV) is a table driven routing protocol created by C. Perkins and P.Bhagwat in 1994. It was based on the distributed Bellman-Ford algorithm. Each Node keeps a list of distances for every destination through each neighbor [1].

This data is kept in a routing table with an order number that shows every access.

The aim of the sequence numbers is to permit nodes to recognize old routes from recent ones to avoid routing loops.

In this routing algorithm each node send refreshes to the routing table regularly, and instantly when notable information is received.

DSDV utilizes two kinds of packets to show its routing table. “Full dump” comprises all obtainable routing information, and “incremental packet” comprises solely data that has altered after the final “Full dump”.

The “Incremental packet” when acquired by a node, the information of the former and the later are checked, then if the packet’s route features a newer sequence number, the equivalent route in the node’s table will be changed.

And if the order numbers are the same and the packet’s route possesses a tinier interval, in this case a packet’s route will substitute the node’s route.

2.3. Wireless Sensor Network Topologies

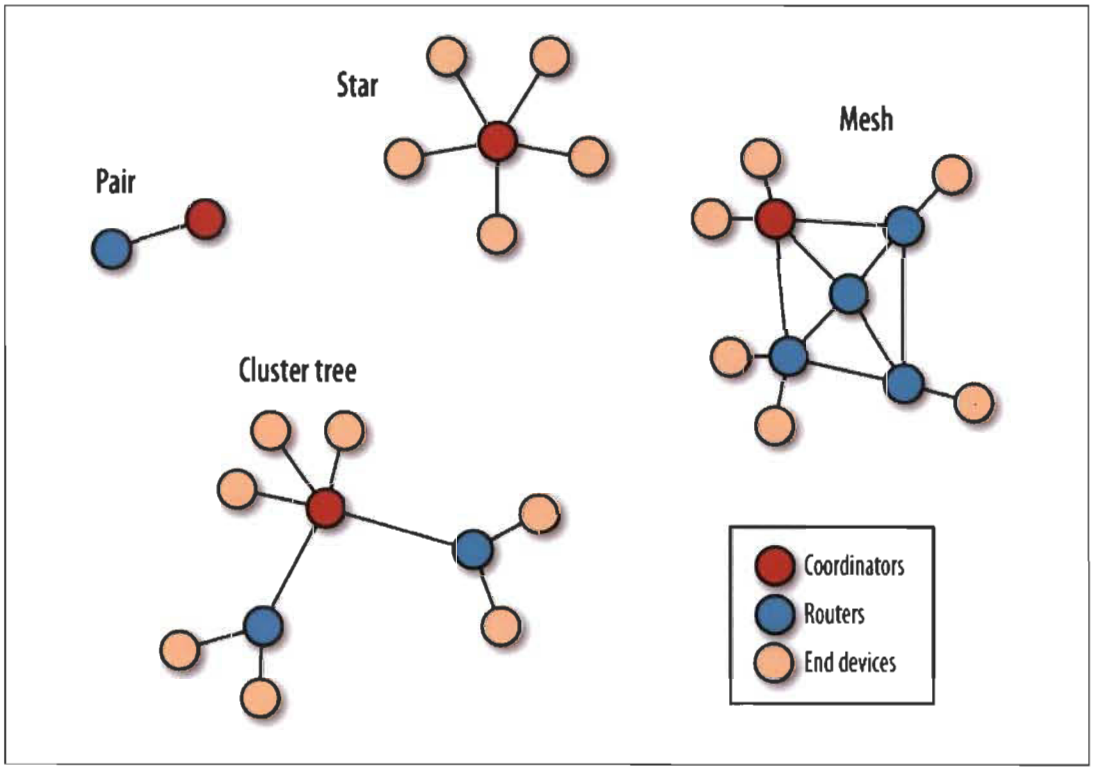


Figure 4 Wireless Sensor Network Topologies [3]

Wireless sensor networks can have several network topologies that dictates the way the radio signals are rationally related together, however their physical disposition can seem dissimilar, the figure above highlights three main topologies: Pair, Mesh and Star [3].

2.3.1 Pair Topology

It is the most basic and simple topology, one node is the coordinator, and the other node is the sink [3].

2.3.2 Star topology

This topology is quite commonly used; the source node or what is also called the coordinator node is in the middle and it connects all other surrounding nodes. Every message in the network must go via the coordinator node that routes them according to the requirements. In addition, the end nodes are not able to exchange information straightforwardly. First, they must communicate with the coordinator node [3].

2.3.3 Mesh Topology

The Mesh topology is also very common, and it utilizes router nodes and the coordinator node. Not like in the case of the star or the pair topology, the coordinator nodes can pass messages to the end nodes via the router nodes [3].

The Coordinator node is actually a particular shape of router, its aim is to administer the WSN. Besides, it is able to route messages [3].

2.3.4 Cluster tree Topology

In this topology the network is divided into clusters and routers are considered like the backbone of the network, and the end nodes grouped around each router [3].

2.4 Related Work

To find out the most efficient rooting protocol for Vehicular Ad-hoc Network A comparison between three rooting protocols (AODV, DSDV, DSR) was done, the comparison metrics were: average throughput, delay, PDR and energy consumed [15]. DSDV had the worst performance and AODV reached the highest throughput and DSR showed a good performance with the lowest delay [15]. Three rooting protocols (AODV, OLSR, ZRP) are compared to find out the security issues of Zigbee (IEEE 802.15.4) standard against what is called the wormhole attack [16].

And the metrics utilized were end to end delay, throughput, Packet delivery ratio PDR and energy consumption and they used Qualnet Simulator 5 as a simulation tool [16].

Assesses the precision estimating the distance between the source and destination nodes for AODV routing protocol in MANET using statistical and neural network models [17]. And they concluded that neural networks outperformed other models such as statistical models and ARIMA model [17].
Highlighted the security aspect of the AODV routing protocol in VANET, they tried to improve the AODV algorithm in order to overcome the Black Hole Attack [18]. Thus an algorithm is suggested that can foster the AODV's security by detecting the black Hole Attacks. Source node saves the route responses in a table [18].

[19] Focused on analyzing the routing performance in the context of robotics. They performed a profound performance comparison between numerous ad hoc routing protocols.

[20] Numerous routing approaches were compared in wireless mesh networks. they attempted to boost routing algorithms and connection metrics
In most of the cases not the best route is chosen due to the routing protocol's issues, interflow interference or inaccurate link metric design [20].

[21] Focused on analyzing the routing protocols for the emergency perspective, many routing approaches were compared such as proactive and reactive routing protocols, to find out which ones are the best for the emergency case scenario.

A metropolitan zone was selected, and three routing protocols were compared AODV, CBRP and DSDV via NS2 network simulator [21].

[22] Compared between performance metrics of AODV and DSDV routing protocols for Mobile Ad hoc Network, Throughput, PDR and routing overhead of both routing protocols were analyzed. [22] came up to the conclusion that, for the scenarios analyzed, the AODV's Performance is superior than DSDV.
However they showed that the performance of AODV decreases significantly when it faces the black hole attack. And they proposed a modified AODV algorithm that can deal better with that kind of attacks [22].

[23] Compared between performance metrics of DSR, AODV, OLSR and DSDV routing protocols for MANET (Mobile Ad hoc Network), Throughput, PDR, delay and routing overhead of the above routing protocols were analyzed via NS2.

[24] proposed a modified version of AODV named irresponsible (iAODV). This routing technique is a probabilistic forwarding suggested by [25] .

[24] found out that iAODV had better performance than AODV in terms of overhead traffic, as it decreased it considerably throughout the route discovery stage.

Chapter 3 : Simulation Tools and environment

NS2 or Network Simulator 2, is basically an event driven simulation software that is widely utilized in experimenting communication networks.

NS2 offers the ability of simulation of both wired and wireless networks as well as their specific protocols [14].

NS2 was created in 1989, and since it is very flexible and composed by several units that can be added to it, NS2 has become widely well-liked in the networking society [14].

Cornell University and the university of California contributed significantly in the evolution of the Network Simulator and in 1995 DARPA founded the improvement of it [14].

Recently the National Science Foundation has become a contributor in the progress of NS2.

3.1 Basic Architecture of NS2

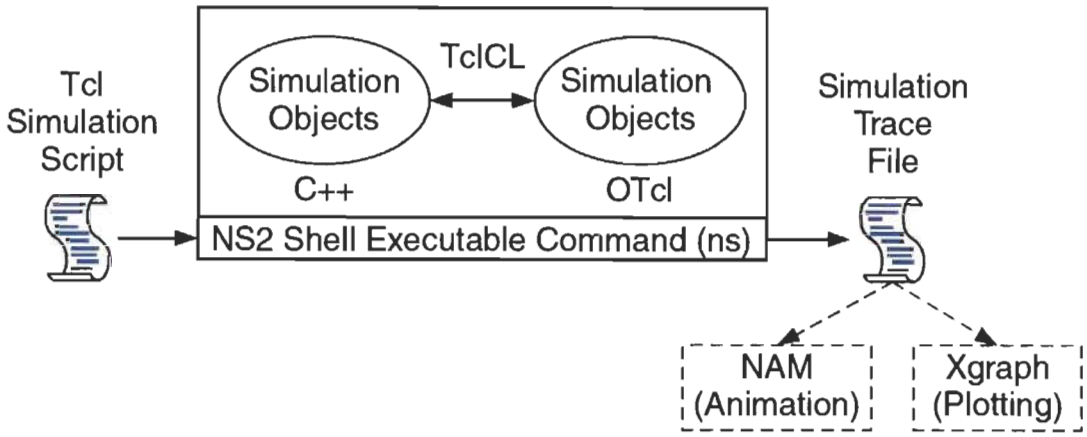


Figure 5 Fundamental Architecture of the Network Simulator NS [14]

This figure above shows the fundamental architecture of NS2. NS2 affords a command named "ns" to run the TCL script.

For exemple to execute a TCL Script named "Exemple1.Tcl", we have to write the following commend "ns Exemple1.Tcl"

Then, if the commend is executed properly, a simulation trace file will be generated, and it is essential to plot a graph and to analyse the network behavior.

NS2 is founded around two main programming languages C++ and OTcl (Object oriented tool command languge). C++ describes the inside mechanism of the

simulation objects, the oTcl designates the outside mechanism such as planning discrete events, building and designing the objects [14].

TclCL connects between the two programming languages (OTcl and C++). Variables defined in a OTcl are linked to a C++ objects. These variable are actually a string in the OTcl and does not hold any role, however the role is described in the linked C++ object. In OTcl, variables behaves as an interface that communicates with users and others OTcl objects [14].

3.2 Installation

NS2 is an open source simulation tool, and it can be downloaded free of charge from its official website ns.com. Even though it has been established in the UNIX Ecosystem, NS2 is able to work on several Operating systems such as Unix, Mac and Windows.

In our case we decided to install NS2 version 2.35 on Linux Ubuntu 12.04 LTS because it works very well on this OS.

Table 3.1 list of Steps to install NS2

Steps	Description
1- Sudo apt-get update	To install necessary updates for the operating system
2- Sudo apt-get install build essential autoconf automake libxmu-dev	To install the library called "libxmu-dev" on the Operating system,
3- Download NS2.35	Download the NS2 zipped file
4- tar zxvf ns-allinone-2.35.tar.gz	Unzip the NS2 zipped file
5- cd ns-allinone-2.35	Access the directory of NS2
6- ./install	Install NS2
7- ./validate	Check if every component is installed properly

3.3 Directories and convention

Now NS2 is installed in directory nsallinone-2.35. The figure below illustrats the directory structure under nsallinone-2.35.

As it is shown in the figure below, they are four directory levels. Firstly, the nsallinone-2.35 directory, then in level two we find the NS2 simulation modules and the TclCL classes, next in the level three there is modules in the interpreted hierarchy and finaly in the fourth level they are the frequently utilized modules.

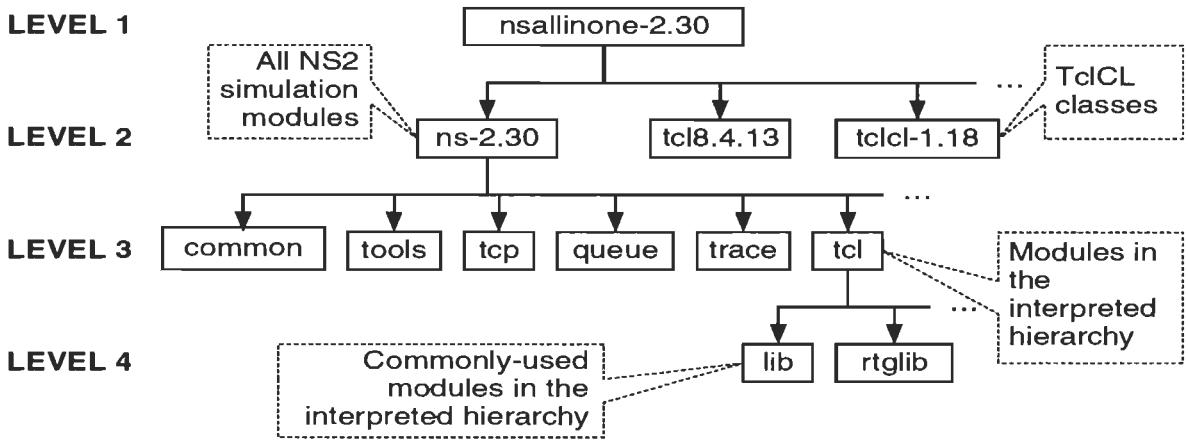


Figure 6 directory structure under NS2 all in one Package [14].

3.4 Running NS2 Simulation

3.4.1 NS2 Program invocation

As NS2 Is now installed, it can be invoced using this commend "NS [filename.tcl] [arg] "

[filename.tcl] [arg] are not compulsory arguments. In that, if the "NS" command did not receive an argument, an NS2 domain will be invoked and NS2 will be ready to execute orders as soon as they are written. And if in addition to the "NS" commend the argenment "[filename.tcl]" will be given, NS2 will execute the whole TCL script.

3.4.2 Main NS2 simulation steps

Table 3.2 Main NS2 simulation steps

Simulation Steps	Description
1- Design	Design of the simulation scenario
2- Write the program	Write the TCL script
3- Run, test and reconfiguration	Run the TCL script
4- Post simulation Processing	Analyse the trace file (out.tr) in order to understand the network behavior.

The table above resumes the main four steps of NS2 simulation, firstly in the design step the aim and the puspose of the network should be fixed as long as the network performance and the configuration, secondly the TCL script should be written in accordance with the Network design, then in the third step the TCL script should be runned to test it and to reconfigure it if necessary. And finally the trace file (out.tr) should be analysed to describe the network behavior and to critisize its performance.

3.5 A Simulation Exemple

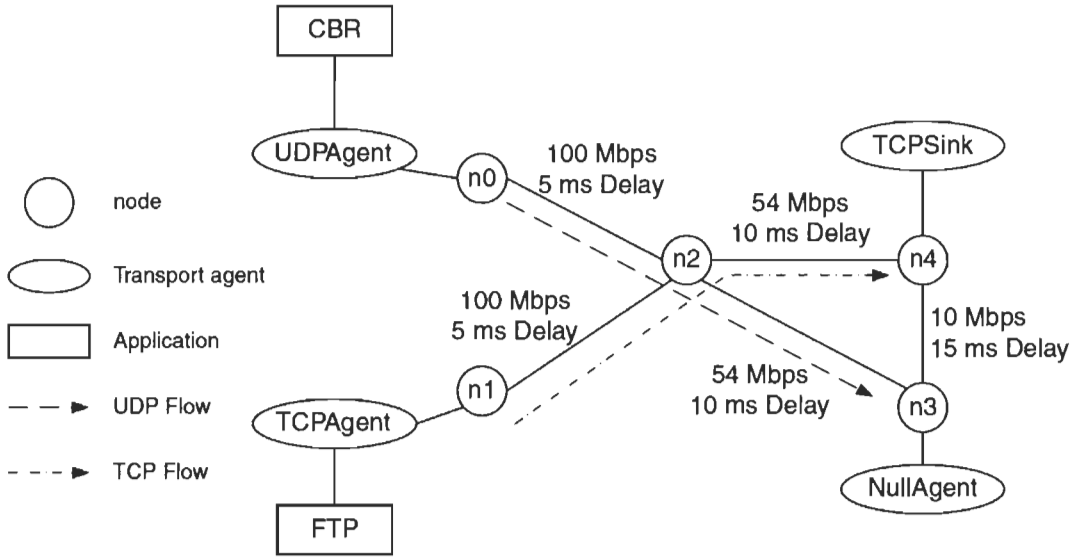


Figure 7 A first Network topology [14]

The figure above highlights a network design formed by five nodes. As it is shown in the figure, n0 transmit CBR traffic to node n3 and node n1 sends FTP traffic to n4.

In NS2 point of view CBR and FTP traffic source are transmitted by UDP and TCP protocols through what is called "UDP agent" and "TCP agent". Whilst the receiver are correspondingly "Null Agent" for UDP and "TCP Sink agent" for TCP.

```
# myfirst_ns.tcl
# Create a Simulator
1  set ns [new Simulator]

# Create a trace file
2  set mytrace [open out.tr w]
3  $ns trace-all $mytrace

# Create a NAM trace file
4  set myNAM [open out.nam w]
5  $ns namtrace-all $myNAM

# Define a procedure finish
6  proc finish { } {
7        global ns mytrace myNAM
8        $ns flush-trace
9        close $mytrace
10       close $myNAM
11       exec nam out.nam &
12       exit 0
13 }

# Create Nodes
14 set n0 [$ns node]
15 set n1 [$ns node]
16 set n2 [$ns node]
17 set n3 [$ns node]
18 set n4 [$ns node]

# Connect Nodes with Links
19 $ns duplex-link  $n0 $n2 100Mb 5ms DropTail
20 $ns duplex-link  $n1 $n2 100Mb 5ms DropTail
21 $ns duplex-link  $n2 $n4 54Mb 10ms DropTail
22 $ns duplex-link  $n2 $n3 54Mb 10ms DropTail
23 $ns simplex-link $n3 $n4 10Mb 15ms DropTail
24 $ns queue-limit $n2 $n3 40
```

Figure 8 a TCL script part 1 [14]

```
#  Create a UDP agent
25 set udp [new Agent/UDP]
26 $ns attach-agent $n0 $udp
27 set null [new Agent/Null]
28 $ns attach-agent $n3 $null
29 $ns connect $udp $null
30 $udp set fid_ 1

#  Create a CBR traffic source
31 set cbr [new Application/Traffic/CBR]
32 $cbr attach-agent $udp
33 $cbr set packetSize_ 1000
34 $cbr set rate_ 2Mb

#  Create a TCP agent
35 set tcp [new Agent/TCP]
36 $ns attach-agent $n1 $tcp
37 set sink [new Agent/TCPSink]
38 $ns attach-agent $n4 $sink
39 $ns connect $tcp $sink
40 $tcp set fid_ 2

#  Create an FTP session
41 set ftp [new Application/FTP]
42 $ftp attach-agent $tcp

#  Schedule events
43 $ns at 0.05 "$ftp start"
44 $ns at 0.1  "$cbr start"
45 $ns at 60.0 "$ftp stop"
46 $ns at 60.5 "$cbr stop"
47 $ns at 61 "finish"

#  Start the simulation
48 $ns run
```

Figure 9 a TCL script part 2 [14]

Chapter 4 : Design and Implementation of the Scenarios

The purpose of this findings and discussion chapter is to present the key relative findings of the research besides explaining their meaning, thus some figures and tables will be highlighted in order to show, on one hand, Whether the wireless sensor network in the case of the intelligent lamps scenarios, is working properly or not and on the other hand, to discuss how much the results found out were similar to what was expected form a theoritical point of view.

Tableau 4.1 Resolution and bandwidth of different video Formats [5]

Format	Resolution	Typical BW
QCIF (1/4 CIF)	176x144	260K
CIF	352x288	512K
4CIF	704x576	1 Mb/s
SD NTSC	720x480	Analog, 4.2Mhz
720 HD	1280x720	1-8 mb/s
1080 HD	1080x1920	5-8 mb/s h.264 12+ mb/s mpg2
CUPC	640x480 max	
YouTube	320x240	Flash(H.264)
Skype	Camera limits	128 - 512K+

This Table above shows the Resolution and bandwidth of different video Formats.

4.1 : Scenario 1 - Comparison of mesh topology 50 nodes with different routing protocols (AODV, DSR, DSDV) (Wi-Fi)

Table 4.2 : Scenario 1 Configuration table

Simulation Tool	NS2 Version 2.34
Network dimension	650 X 650
Number of nodes	50 nodes
Simulation Time	100 seconds
Maximum queue	50 packets
Data transfer mode	Direct transmission
Maximum air data rates	IEEE 802.11 : 2000 Kbps (Wi-Fi)
Node mobility	off
Propagation model	Two-ray Ground
Rooting potocols	AODV, DSR, DSDV
Channel Capacity (Maximum Data rate) = (Video Format Bandwidth)	1 Mb/s (Mega bit per second (Mbps))
Packet size	512 Bytes
Transfer Protocol	UDP
Data traffic	CBR
Topology	Mesh

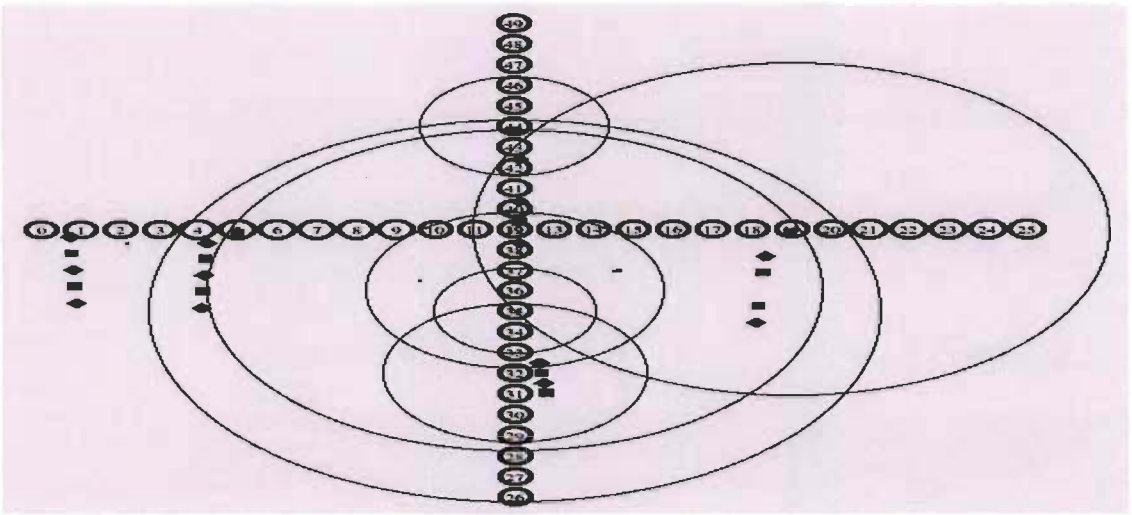


Figure 10 Topology of our scenario 1 (50 nodes) with different routing protocols (AODV, DSR, DSDV) (Wi-Fi)

As it is shown in the figure above the topology of the first scenario is composed by 50 nodes representing the smart lights in the street they are spread on a surface of 650 meter X 650 meter. The distance between smart lights is 25 meters (like in real life) and as it is outlined In the table above, three routing protocols is going to be tested (AODV, DSDV, DSR) in the same condition and in the same Network configuration parameters in that, the Simulation Time is going to be 100 seconds, the maximum queue allowed 50 packets, the data transfer mode direct transmission, the maximum air data rates (IEEE 802.11: 2000 Kbps (Wi-Fi)) Node mobility will be off and the propagation model is Two-ray Ground.

In addition the Packet size will be 512 Bytes, Transfer Protocol will be the UDP since it is commonly used for video transmission, and for data traffic we are

going to utilize the CBR as it is frequently used for streaming multimedia content. And because many nodes are interconnected the topology of our network is going to be the Mesh topology.

4.1.1 Throughput Mesh topology 50 nodes with different routing protocols (AODV, DSR, DSDV) (Wi-Fi)

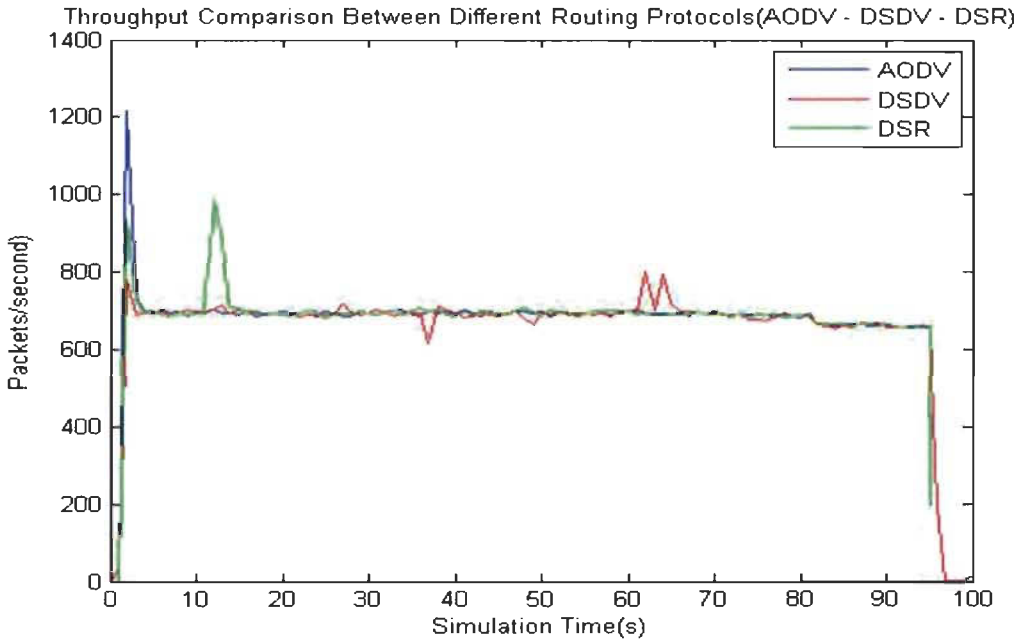


Figure 11 Throughput comparison between Different Routing Protocols (AODV, DSDV, DSR) - Our Result

In this scenario we are going to compare these performance metrics : Throughput, Jitter and End to end delay of Mesh topology (50 nodes) with different routing protocols (AODV, DSR, DSDV) in order to find out witch routing protocol is performing better.

In the figure above we compared the throughput of different routing protocols of nodes AODV, DSR and DSDV.

As it is shown in the table above, The setup for the simulation scenario functioning at 1 Mbps (Data rate) for the three senarios DSDV as well as AODV and DSR.

As it is shown in the figure above, we notice that the highest throughput when we utilized the AODV routing protocol, in the first seconds of the simulation, it plummeted to reach about 700 Packets per second.
Then by reaching the 4^{th} second all the routing protocls were quite steady in around 700 packets per second, whilst by the 12^{th} second DSR's throughput rose sudinly to reach around 1000 packets per second.

DSDV's throughput showed less stability than the others since by reaching the 35^{th} second it decreased to reach 600 Packets per second then between 60 and 70 seonds it fluctuated between 800 and 700 Packets per second.

When there is a big traffic, the collision rate escalates and this is how the throughput of the system is affected. We observe that the highest throughput when When AODV was utilized was around 1200 packets per second at the third second and during most of the time of the simulation, the throughput remained steady at around 700 Packets per second until the end of the simulation.

4.1.2 Jitter Mesh topology 50 nodes with different routing protocols (AODV, DSR, DSDV) (Wi-Fi)

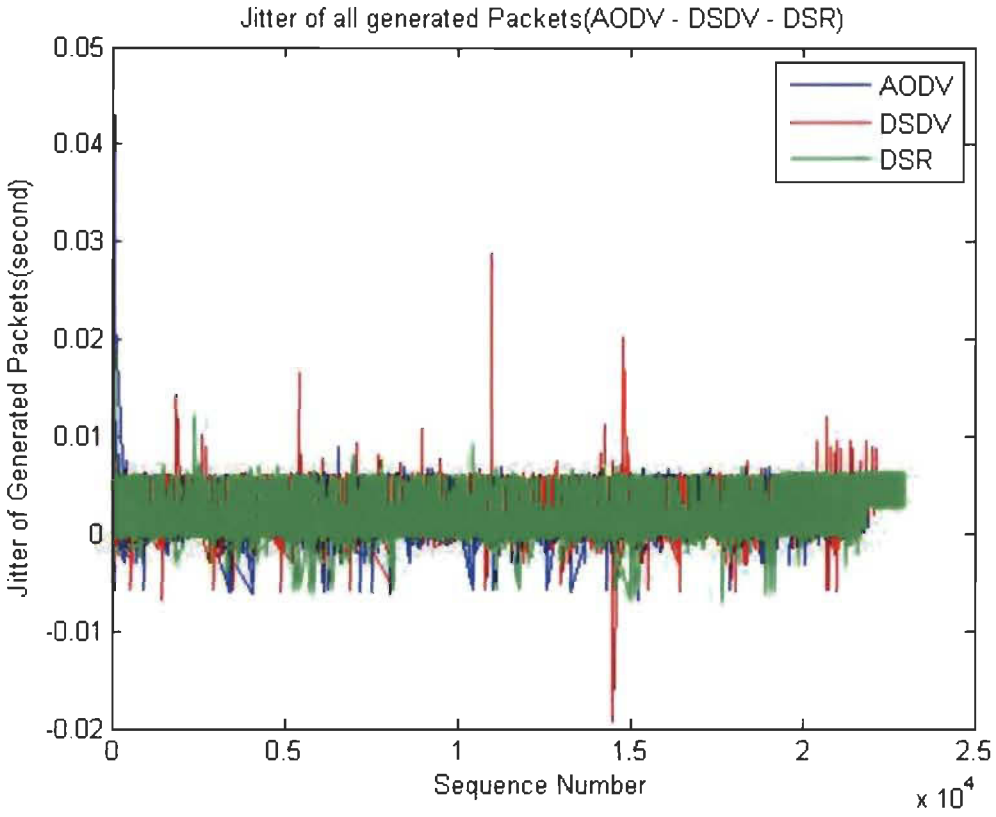


Figure 12 Jitter comparison between Different Routing Protocols (AODV, DSDV, DSR) - Our Result

The difference in the delay of received packets Is called jitter.

From a sending point of view, packets are transmitted in a steady flow and there is a time frame between the different packets.

Various factors such as network congestion, inappropriate queuing, the routing protocol used, might cause some issues like the instability of the network data flow, or the fluctuating variation of the delay between every packet.

This figure above shows the collective spreading of the jitter throughout the simulation time for different routing protocols (AODV, DSDV and DSR) using the same number of nodes 50 nodes.

We notice that in the scenario of the AODV routing protocol, at the beginning of the simulation, the jitter rose drastically and reach more than 0.04 seconds, then it steadied to become quite similar to the other two scenarios (DSDV and DSR routing protocol), and to fluctuate between 0 and 0.01 seconds for most of the time.

Moreover, comparing DSR with other routing protocols it has the most stable jitter, however DSDV has the most unstable jitter amomg the three routing protocols, since after about 5000 sequences the jitter reached around 0.02 seconds and after 12000 sequences it reached about 0.03 seconds and after about 15000 sequences it reached 0.02 seconds and by the end the sequences it 50tilize5050d between 0 and 0.01 seconds.

4.1.3 End to End Delay Mesh topology 50 nodes with different routing protocols (AODV, DSR, DSDV) (Wi-Fi)

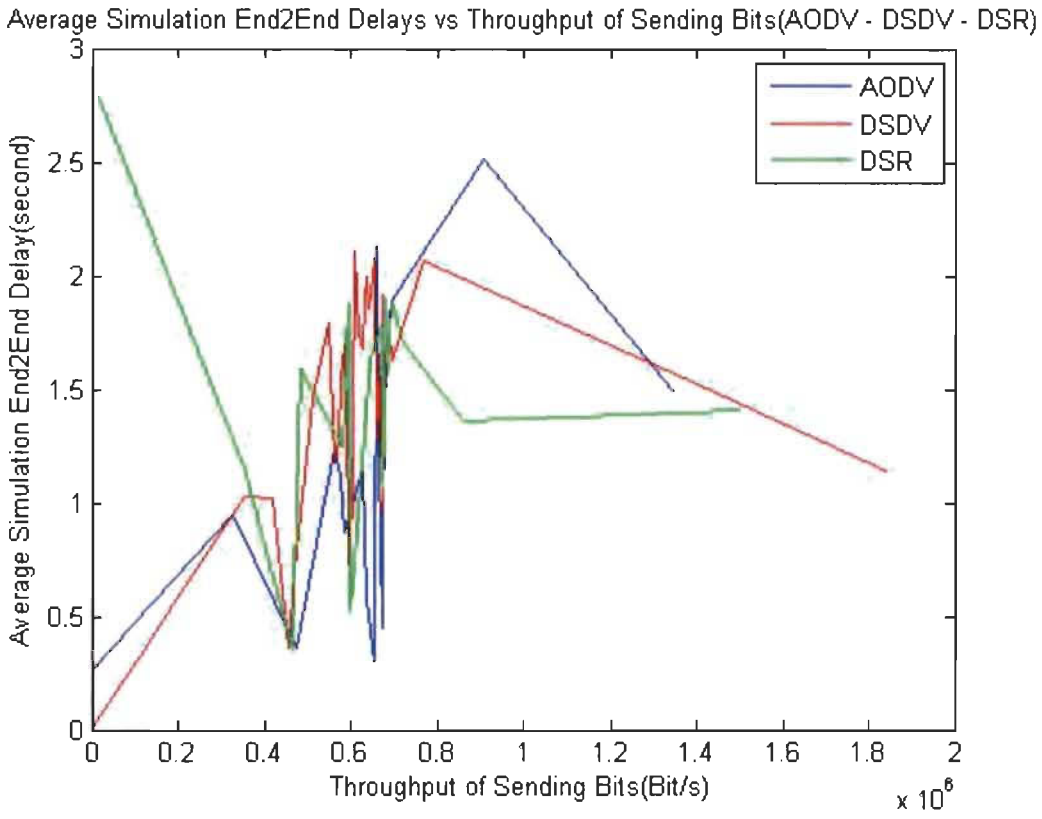


Figure 13 End to End Delay comparison between Different Routing Protocols (AODV, DSDV, DSR) - Our Result

The end-to-end delay is the time duration for a packet to be sent plus the time duration to obtain an acknowledgment, in other words, the delay involves the data transmission time between the two spots of signal.

In the figure above we notice that the overall end to end delay of the DSR routing protocol scenario was better when we compare it to the other routing protocols.

The Analyze of the results of different routing protocols AODV, DSDV, DSR and the same topologies and the same number of nodes shows that the end to end network performance depends on the routing protocol that has been used.

4.2 : Scenario 2 - Mesh Topology 50 nodes (Wi-Fi) Comparaison between different video formats (CIF, 4CIF, QCIF) with DSDV routing Protocol

Tableau 4.3 Scenario 2 Configuration table

Simulation Tool	**NS2 Version 2.34**
Network dimension	650 X 650
Number of nodes	50 nodes
Simulation Time	100 seconds
Maximum queue	50 packets
Data transfer mode	Direct transmission
Maximum air data rates	IEEE 802.11 : 2000 Kbps (Wi-Fi)
Node mobility	off
Propagation model	Two-ray Ground
Rooting potocols	DSDV
Channel Capacity (Maximum Data rate) = (Video Format Bandwidth)	260 Kb/s, 512 kb/s, 1 Mb/s (Mega bit per second (Mbps))
Packet size	512 Bytes
Transfer Protocol	UDP
Data traffic	CBR
Topology	Mesh

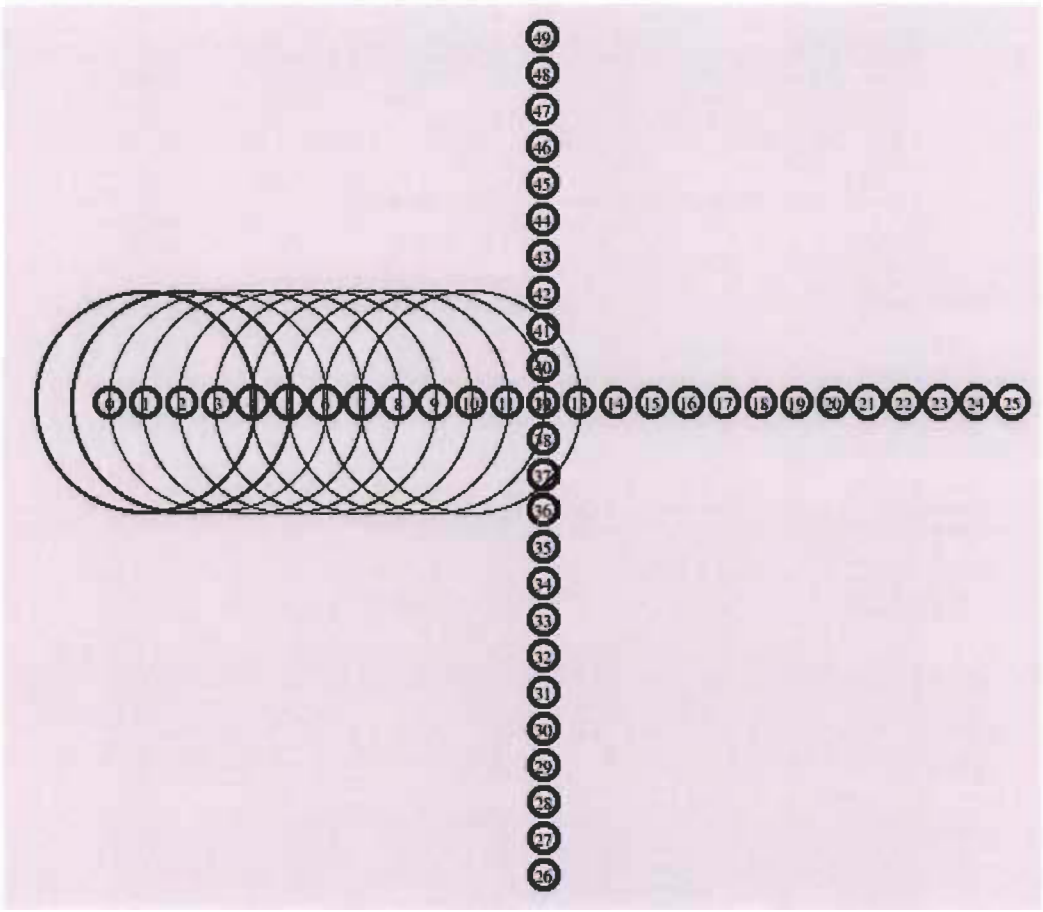


Figure 14 Topology of the scenario 1 (50 nodes) with different video formats (CIF, 4CIF, QCIF ...) with DSDV routing Protocol - Our Result

As it is shown in the figure above the topology of the first scenario is composed by 50 nodes representing the smart lights in the street they are spread on a surface of 650 meter X 650 meter. The distance between smart lights is 25 meters (like in real life) and as it is outlined In the table above, three video formats is going to be tested (CIF, 4CIF, QCIF) in the same condition and in the same Network configuration parameters in that, the Simulation Time is going to be 100 seconds, the routing protocols is going to be DSDV, the maximum queue allowed 50 packets, the data transfer mode direct transmission, the maximum air

data rates (IEEE 802.11: 2000 Kbps (Wi-Fi)) Node mobility will be off and tge Propagation model is Two-ray Ground.

In addition the Packet size will be 512 Bytes, Transfer Protocol will be the UDP since it is commonly used for video transmission, and for data traffic we are going to utilize the CBR as it is frequently used for streaming multimedia content. And because many nodes are interconnected the topology of our network is going to be the Mesh topology.

Tableau 4.4 Scenario 2 Results

	Video Resolution	Packet sent (generated)	Packet Received	(PDR) Packet Delivery Ration Packet Received/ Packet sent	Average End to End Delay (s)
DSDV Video format Bandwidth 4CIF (1Mbps)	704X576	65147	-36799	-0.564	1.57
DSDV Video format Bandwidth CIF (512 Kbps)	352X288	64379	19120	0.297	1.60
DSDV Video format Bandwidth QCIF (260 Kbps)	176X144	62110	46242	0.744	1.49

4.2.1 Throughput of the Mesh Topology 50 nodes (Wi-Fi)

Comparaison between different video formats with DSDV routing Protocol

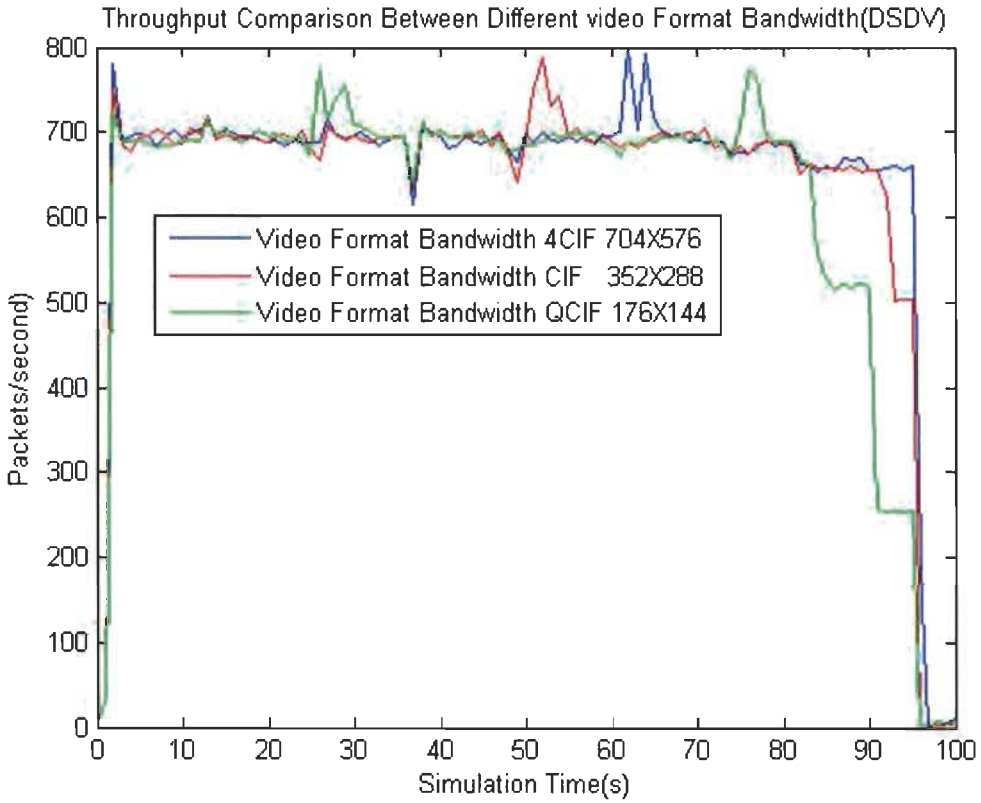


Figure 15 Throughput : Mesh Topology 50 nodes (Wi-Fi) Comparaison between different video formats with DSDV routing Protocol - Our Result

Analyzing Throughput is very popular among network parameters, to find out how the network is performing. Not like in the case of transmitting text messages, video application might require much more bandwidth.

In the figure above we compared the throughput of different video formats 4CIF (704X576) CIF (352X288) and QCIF (176X144).

The setup for the simulation scenario functioning at these following data rates 260 Kb/s, 512 kb/s, 1 Mb/s for these three senarios with the following video format 4CIF (704X576), CIF (352X288) and QCIF (176X144).

As it is shown in the figure above, we noticed that, with the 4CIF video format, the highest throughput reached was about 800 packets per second and the lowest throughput reached was 600 packets per second and, it steadied to about 700 packets per seconds for most of the simulation time.

On the other hand, with the CIF video format, the highest throughput reached was less than 800 packets per second and the lowest throughput reached was 650 packets per second and, it steadied to about 700 packets per seconds for most of the simulation time.

Moreover, with the QCIF video format, the highest throughput reached was less than about 780 packets per second and the lowest throughput reached was 600 packets per second and, it steadied to about 700 packets per seconds for most of the simulation time.

Furthermore, for all the video formats, In the beginning of the simulation, the throughput increased dramatically between 0 and 4 seconds until it reached about 700 Packets per seconds, then after the fourth second it fluctuated steadily

until it reached the end of the simulation, that fluctuation in the throughput is due to the rises and declines in collusion and traffic at specific time. When there is a big traffic, the collision rate escalates and this is how the throughput of the system is affected.

4.2.2 Jitter Mesh Topology 50 nodes (Wi-Fi) Comparaison between different video formats with DSDV routing Protocol

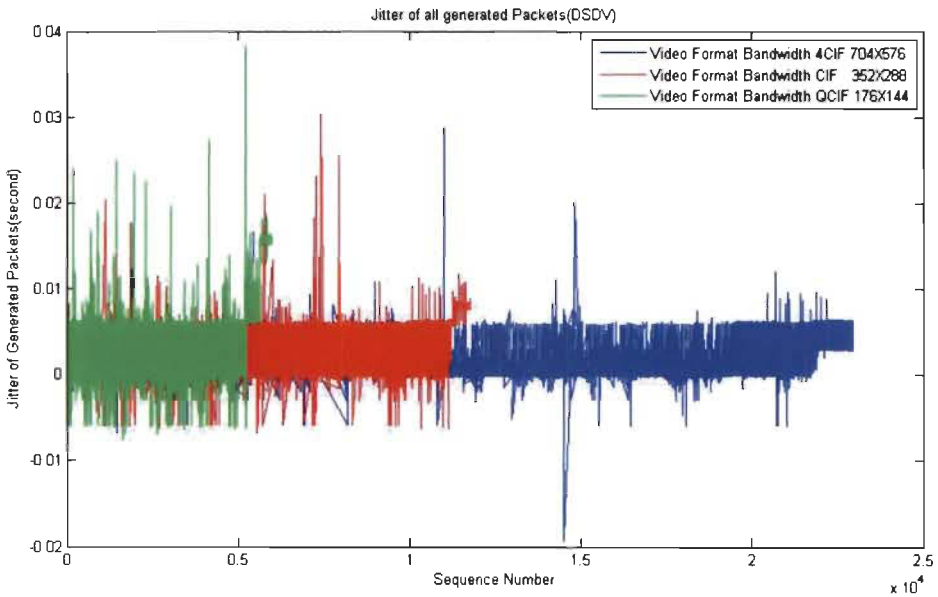


Figure 16 Jitter : Mesh Topology 50 nodes (Wi-Fi) Comparaison between different video formats with DSDV routing Protocol - Our Result

The difference in the delay of received packets is called jitter.

From a sending point of view, packets are transmitted in a steady flow and there is a time frame between the different packets.

Various factors such as network congestion, inappropriate queuing, the routing protocol used, might cause some issues like the instability of the network data flow, or the fluctuating variation of the delay between every packet.

This figure above shows the collective spreading of the jitter throughout the simulation time for different video formats 4CIF (704X576) CIF (352X288) and QCIF (176X144) using the same number of nodes (50 nodes) and the same routing protocol (DSDV).

Moreover, comparing the 4CIF (704X576) topology's scenario with other topologies' scenarios it has the most stable jitter, however QCIF (176X144) topology has the most unstable jitter amomg the three topologies' scenarios, since it fluctuated between about 0.04 seconds and slightly less than 0 seconds, in addition, for the CIF (352X288) topology's scenario, the jitter fluctuated between about 0.03 seconds and slightly less than 0 seconds.

4.2.3 End to End Delay Mesh Topology 50 nodes (Wi-Fi) Comparaison between different video formats with DSDV routing Protocol

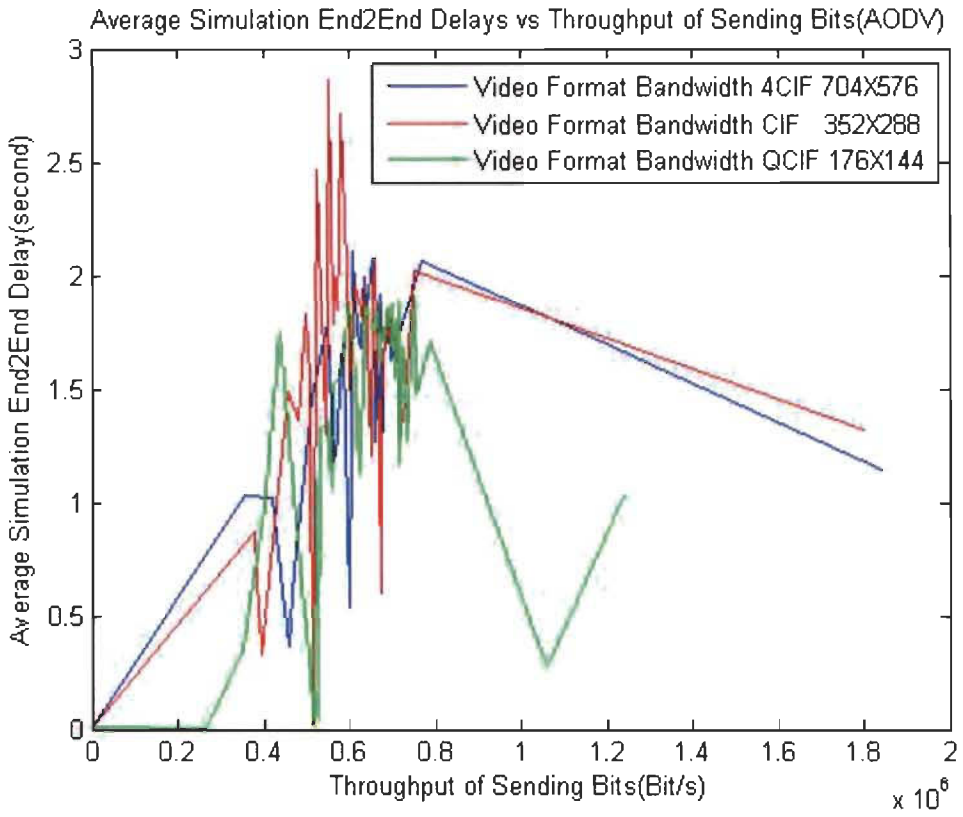


Figure 17 End to End Delay : Mesh Topology 50 nodes (Wi-Fi) Comparaison between different video formats with DSDV routing Protocol - Our Result

The end-to-end delay is the time duration for a packet to be sent plus the time duration to obtain an acknowledgment,

In other words, this delay involves the data transmission time between the two spots of signal.

In the figure above we notice that the overall end-to-end delay of the Video

format QCIF scenario is better when we compare it to the 4CIF and Cif formats.

The Analyze of the results of different video formats 4CIF (704X576) CIF (352X288) and QCIF (176X144), and with the same topologies and the same number of nodes and with same routing protocol, shows that the end to end network performance depends on the video format that has been used.

4.3 : Scenario 3 - Wi-Fi Result vs Zigbee Result 50 nodes Mesh Topology (with AODV)

Tableau 4.5 Scenario 3 Configuration table

Simulation Tool	NS2 Version 2.34
Network dimension	650 X 650
Number of nodes	50 nodes
Simulation Time	100 seconds
Maximum queue	50 packets
Data transfer mode	Direct transmission
Maximum air data rates	IEEE 802.11 : 2000 Kbps (Wi-Fi) IEEE 802.15.4 250 Kbps (Zigbee)
Node mobility	off
Propagation model	Two-ray Ground
Rooting potocols	AODV,
Channel Capacity (Maximum Data rate) = (Video Format Bandwidth)	260 Kb/s (Kilo bit per second (Mbps))
Packet size	512 Bytes
Transfer Protocol	UDP
Data traffic	CBR
Topology	Mesh

1/ Scenario 1 : Topology 50 nodes Comparaison between Wi-Fi and Zegbee results with the AODV routing protocols (For the Transport layer we used UDP CBR) Null) and Using the same video Format (QCIF (Typical Bandwidth 260 Kbps)).

Topology for both scenarios (Wi-Fi and Zigbee) is showm in the figures Below :

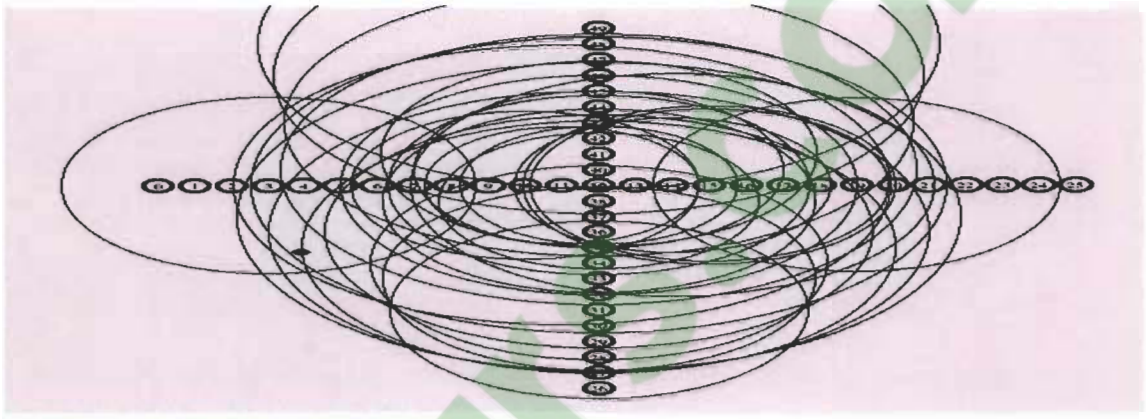


Figure 18 The scennario's topology (Wi-Fi) - Our Result

In this case the video format will be QCIF (1/4 CIF) because it has a bandwidth of 260 kbps. (The closest to the zigbee channel Capacity (250 kbps))

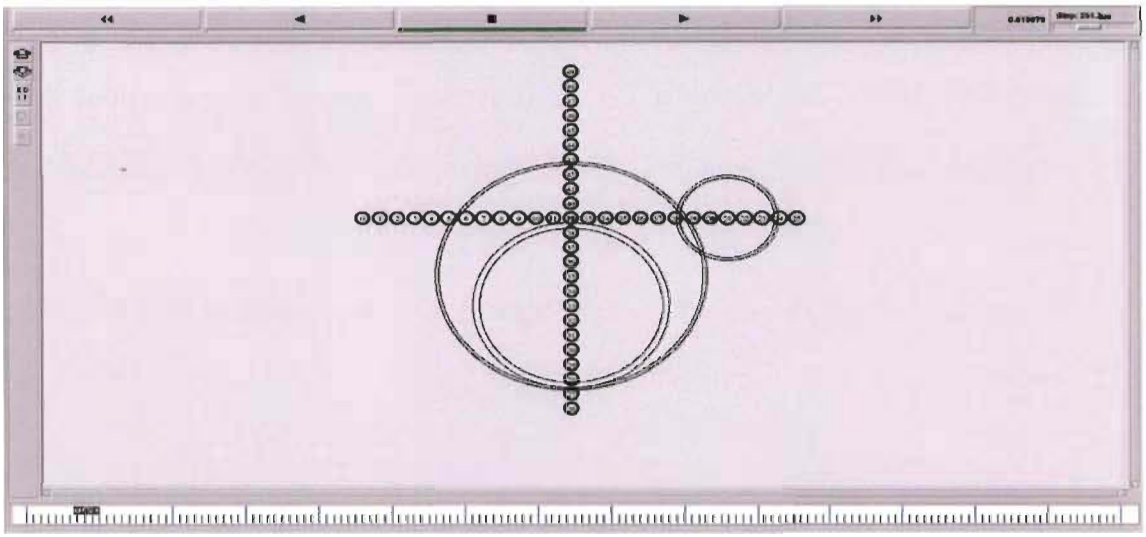


Figure 19 The scennario's topology (Zigbee) - Our Result

As it is shown in the figure above the topology of the first scenario is composed by 50 nodes representing the smart lights in the street they are spread on a surface of 650 meter X 650 meter. The distance between smart lights is 25 meters (like in real life) and as it is outlined In the table above, the routing protocols is going to be (AODV) in the same condition and in the same Network configuration parameters. In that, the Simulation Time is going to be 100 seconds, the maximum queue allowed 50 packets, the data transfer mode direct transmission, the maximum air data rates (IEEE 802.11: 2000 Kbps (Wi-Fi) and IEEE 802.15.4 250 Kbps (Zigbee)) Node mobility will be off and the propagation model is Two-ray Ground.

In addition the Packet size will be 512 Bytes, the Video Format Bandwidth will be 260 Kb/s (Kilo bit per second (Mbps)), the Transfer Protocol will be the UDP since it is commonly used for video transmission, and for data traffic we are going to utilize the CBR as it is frequently used for streaming multimedia content.

And because many nodes are interconnected the topology of our network is going to be the Mesh topology.

4.3.1 Throughput of the Wi-Fi Result vs Zigbee Result 50 nodes Mesh Topology (with AODV)

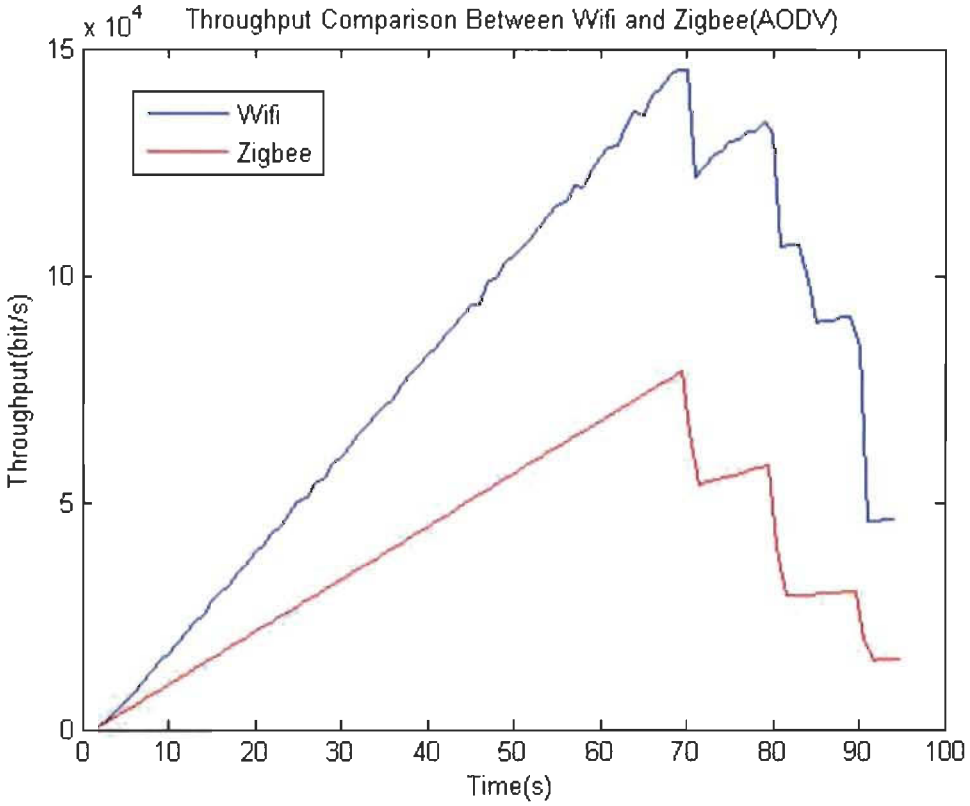


Figure 20 Throughput : Wi-Fi Result vs Zigbee Result 50 nodes Mesh Topology (with AODV)

The setup for the simulation scenario are presented in the table above, data rates 260 Kb/s for these two senarios Wi-Fi and Zigbee with the same number of nodes (50 nodes) and the same Network Topology (Mesh Topology) and with the same routing protocol (AODV).

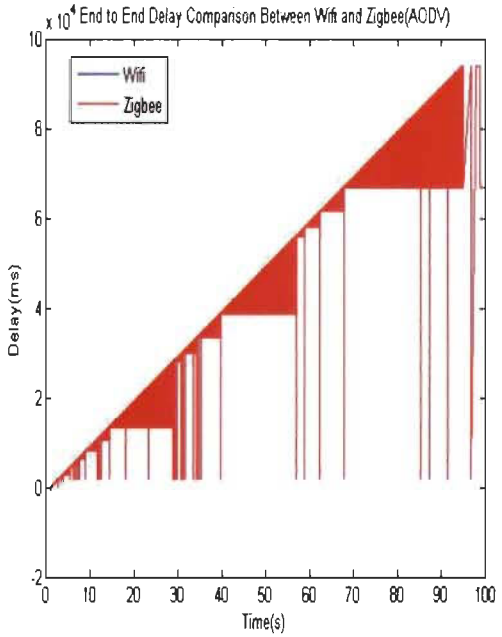
As it is shown in the figure above, we noticed that, with the Wi-Fi 802.11 Protocol, the highest throughput reached was about 150 Kb/s packets per second and the lowest throughput reached was 50 Kb/s, after about 70 seconds, it began decreasing steadly until it reached 50 Kb/s at the end of the simulation time.

On the other hand, with the Zigbee 802.15.4 Protocol, the highest throughput reached was about 70 Kb/s packets per second and the lowest throughput reached was 10 Kb/s, after about 70 seconds, it began decreasing steadly until it reached 10 Kb/s at the end of the simulation time.

Furthermore, for all the video formats, In the beginning of the simulation, the throughput increased dramatically between 0 and 70 seconds until it reached about 150 Kb/s for Wi-Fi 802.11 and 70 Kb/s for Zigbee, then after the seventh second it fluctuated steadily until it reached the end of the simulation, that fluctuation in the throughput is due to the rises and declines in collusion and traffic at specific time.

When there is a big traffic, the collision rate escalates and this is how the throughput of the system is affected. Thus In terms of data rate The Wi-Fi 802.11 protocol showed more capability than the Zigbee 802.15.4 protocol

4.3.2 End to end Delay : Wi-Fi Result vs Zigbee Result 50 nodes Mesh Topology (with AODV)



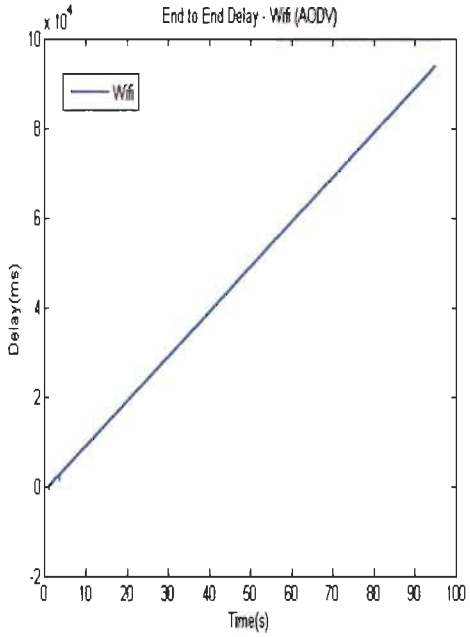


Figure 21 End to End Delay : Wi-Fi vs Zigbee *Figure 22 End to End Delay : Wi-Fi vs Zigbee - Our Results*

The end-to-end delay is the time duration for a packet to be sent plus the time duration to obtain an acknowledgment,

In other words, this delay involves the data transmission time between the two spots of signal.

In the figure above we notice that the overall end to end delay of the both scenarios (Wi-Fi and Zigbee) inceases thoughout time(s) and that is due to the traffic congestion that Inceases during time since new nodes interferes with the

network however with the end-to-end delay of the Zigbee 802.15.4 ptotocol the delay fluctuated between 0 and 90 seconds, there fore It showed a better performance than the Wi-Fi 802.11 protocol.

The Analyze of the results of different Communication protocols Wi-Fi 802.11 and Zigbee 802.15.4 and the same mesh topologies and the same number of nodes (50 nodes) shows that the network performance depends on the Communication protocol that has been used.

4.4 : Scenario 4 - Comparison between topologies (Star Topology, Mesh Topology and line topology) 11 nodes (Wi-Fi) AODV

Tableau 4.6 Scenario 4 Configuration table

Simulation Tool	**NS2 Version 2.34**
Network dimension	850 X 900
Number of nodes	11 nodes
Simulation Time	100 seconds
Maximum queue	50 packets
Data transfer mode	Direct transmission
Maximum air data rates	IEEE 802.11 : 2000 Kbps (Wi-Fi)
Node mobility	Off
Propagation model	Two-ray Ground
Rooting potocols	AODV,
Channel Capacity (Maximum Data rate) = (Video Format Bandwidth)	1 Mb/s (Kilo bit per second (Mbps))
Packet size	1500 Bytes
Transfer Protocol	UDP
Data traffic	CBR
Topology	Mesh / Line / Star

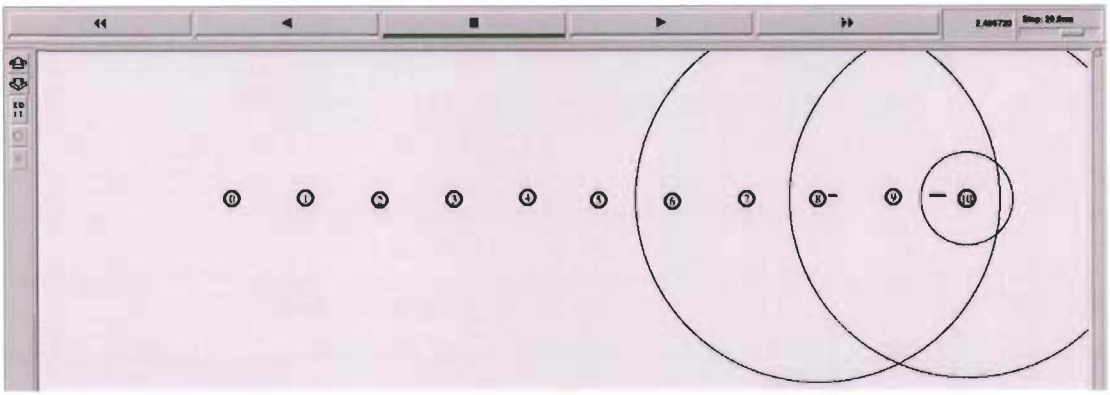


Figure 23 Line Topology 11 nodes (Wi-Fi) AODV – our result

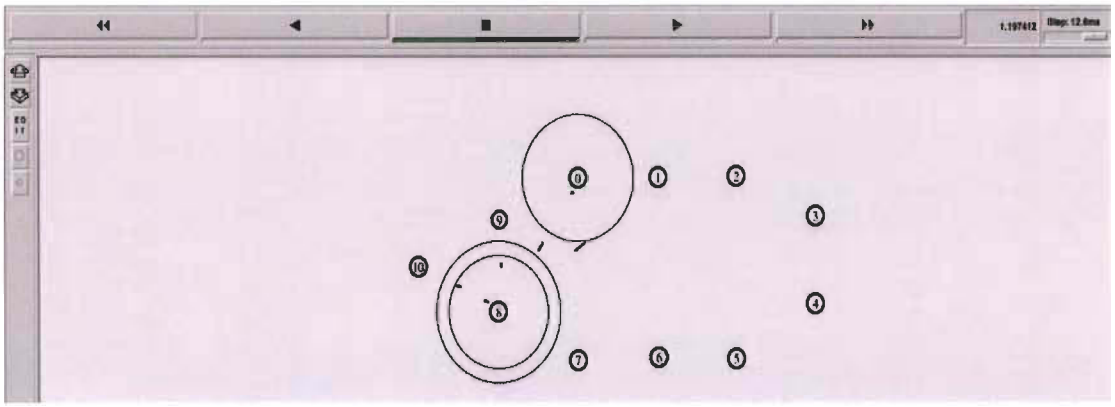


Figure 24 Mesh Topology 11 nodes (Wi-Fi) AODV – our result

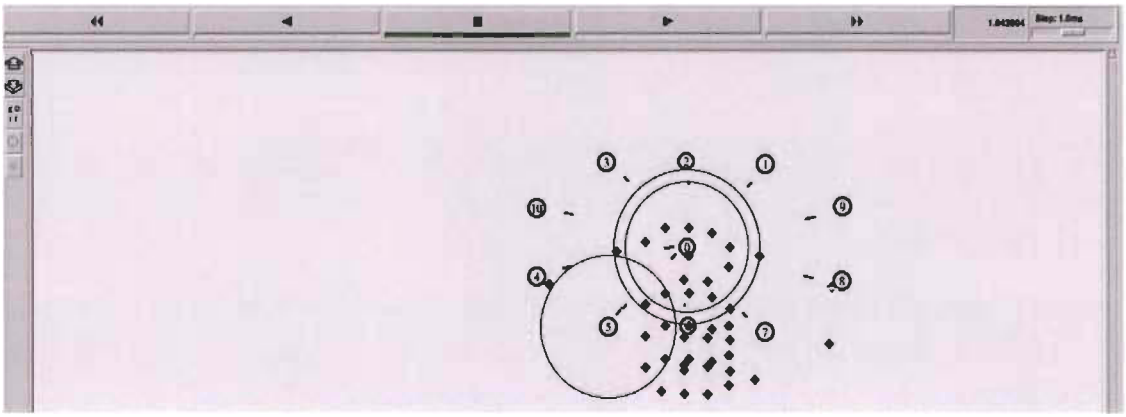


Figure 25 Star Topology 11 nodes (Wi-Fi) AODV – our result

As it is shown in the figure above the topology of the first scenario is composed by 11 nodes representing the smart lights in the street they are spread on a surface of 850 meter X 900 meter. The routing protocols are going to be (AODV) in the same condition and in the same Network configuration parameters. In that, the Simulation Time is going to be 100 seconds, the maximum queue allowed 50 packets, the data transfer mode direct transmission, the maximum air data rates (IEEE 802.11: 2000 Kbps (Wi-Fi) Node mobility will be off and the propagation model is Two-ray Ground.

In addition the Packet size will be 1500 Bytes, the Video Format Bandwidth will be 1 Mb/s (Mega bit per second (Mbps)), the Transfer Protocol will be the UDP since it is commonly used for video transmission, and for data traffic we are going to utilize the CBR as it is frequently used for streaming multimedia content. And three network topologies are going to be tested, the Mesh, Line and star topology.

4.4.1 Throughput Comparison between topologies (Star Topology, Mesh Topology and line topology) 11 nodes (Wi-Fi) AODV

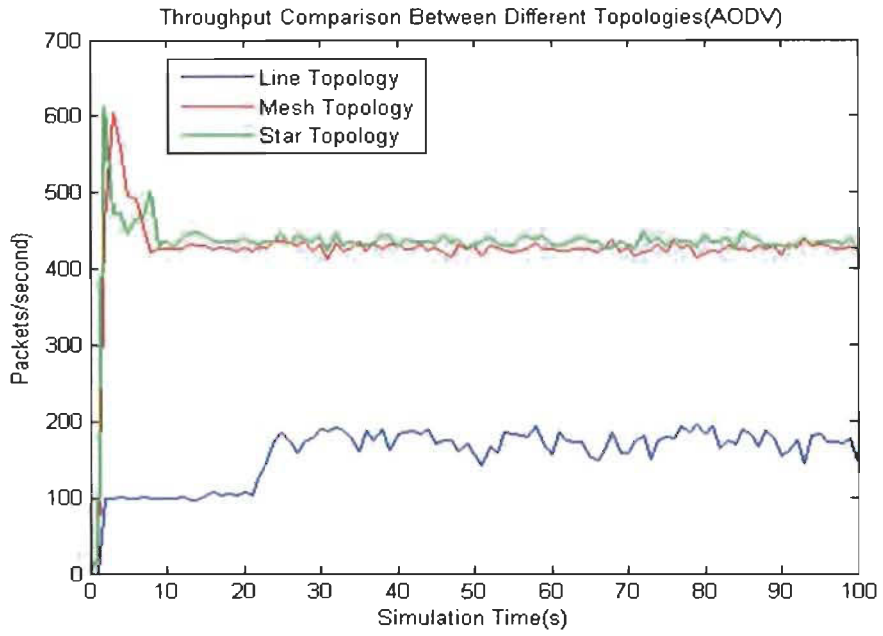


Figure 26 Throughput Comparison between topologies (Star Topology, Mesh Topology and line topology) 11 nodes (Wi-Fi) AODV. - Our Result

In the figure above we compared the throughput of different network topology Line, Mesh and Star.

The setup for the simulation scenario are presented in the table above, the data rate is fixed to 1500 b/s, and the number of nodes is 11 nodes, the Transfer Protocol is UDP and CBR was used as the Data traffic and the routing protocol is AODV.

As it is shown in the figure above, we noticed that, with the star topology, the highest throughput reached was about 610 packets per second and the lowest throughput reached was 450 packets per second and, it stabilized to that throughput for most of the simulation time.

In addition, as it is presented in the figure above, we noticed that, with the Mesh topology, the highest throughput reached was about 600 packets per second and the lowest throughput reached was about 430 packets per second and, it stabilized to that throughput for most of the simulation time.

Besides, with the line topology, the highest throughput reached was about 200 packets per second then, it fluctuated between 150 and 200 packets per seconds for most of the simulation time.

Additionally, for all the network topologies, In the beginning of the simulation, the throughput increased dramatically between 0 and around 3 seconds until it reached about 100 Packets per seconds for line topology and slightly more than 600 packet per second for star and Mesh topologies, then after the fourth second it fluctuated steadily until it reached the end of the simulation, that fluctuation in the throughput is due to the rises and declines in collusion and traffic at specific time.

4.4.2 Jitter Comparison between topologies (Star Topology, Mesh Topology and line topology) 11 nodes (Wi-Fi) AODV

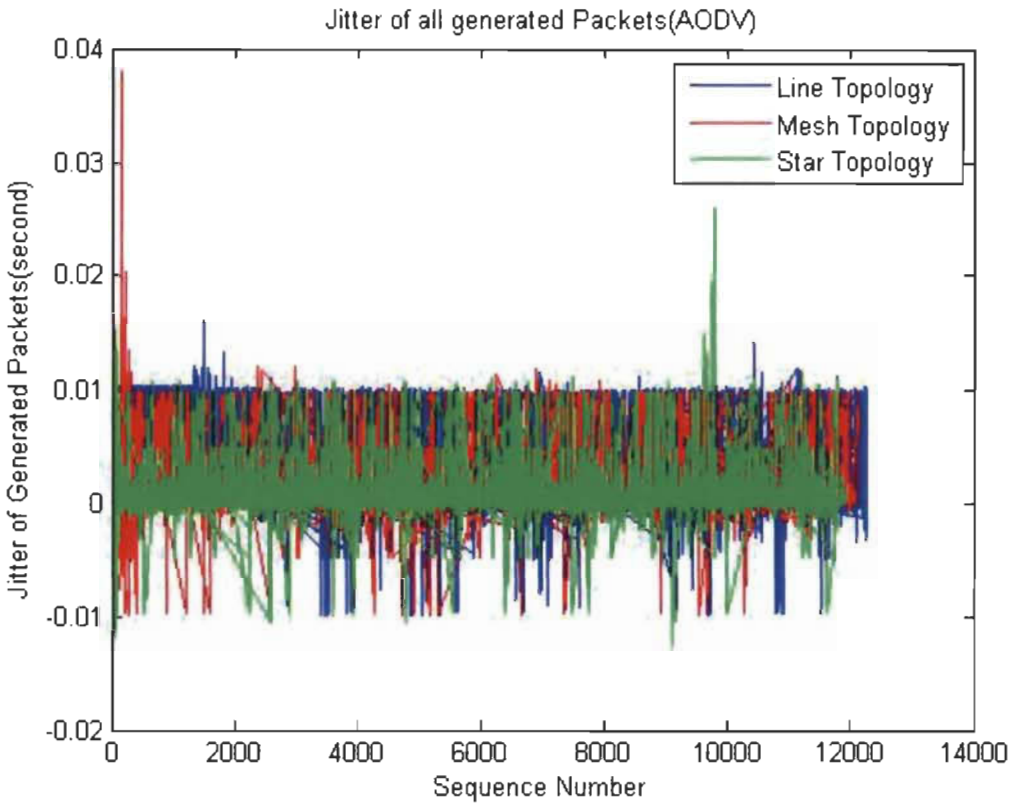


Figure 27 Jitter Comparison between topologies (Star Topology, Mesh Topology and line topology) 11 nodes (Wi-Fi) AODV. - Our Result

The difference in the delay of received packets is called jitter. From a sending point of view, packets are transmitted in a steady flow and there is a time frame between the different packets.

Various factors such as network congestion, inappropriate queuing, the routing protocol used, might cause some issues like the instability of the network data flow, or the fluctuating variation of the delay between every packet.

This figure above shows the collective spreading of the jitter throughout the simulation time for different network Topologies (Line, Mesh and Star) using the same number of nodes (11 nodes) and the same routing protocol (AODV).
We notice that in the scenario of the Mesh topology, at the beginning of the simulation, the jitter rose drastically and reach around 0.04 seconds, then it steadied to become quite similar to the other two scenarios (Line and Star topology), and fluctuate between 0 and 0.01 seconds for most of the time .

Moreover, comparing the line topology's scenario with other topologies' scenarios it has the most stable jitter, however star topology has the most unstable jitter amomg the three topologies' scenarios, since after about 9000 sequences the jitter reached less than 0.01 seconds and after about 9500 sequences it reached about 0.03 seconds and by the end of the simulation it utilized between 0 and 0.01 seconds.

4.4.3 End to End Delay Comparison between topologies (Star Topology, Mesh Topology and line topology) 11 nodes (Wi-Fi) AODV

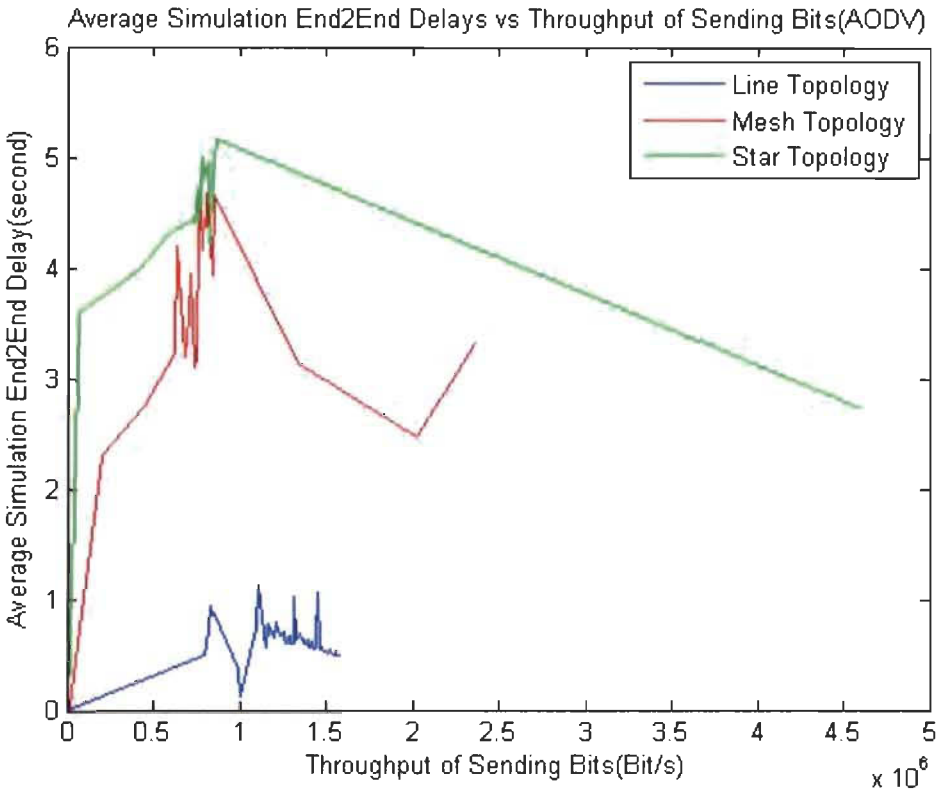


Figure 28 End to End Delay Comparison between topologies (Star Topology, Mesh Topology and line topology) 11 nodes (Wi-Fi) AODV. - Our Result

The end-to-end delay is the time duration for a packet to be sent plus the time duration to obtain an acknowledgment, In other words, this delay involves the data transmission time between the two spots of signal.

In the figure above we notice that the overall end to end delay of the line topology's scenario is better when we compare it to the Mesh and Star topology's scenario.

The star topology has the worst delay, and that is due to the congestion of the networks, since in the Star Topology's scenario all the nodes are communicating

directly with the source node, as well as in the Mesh Topology's scenario many destination nodes are communicating with several routing nodes however in the line topology's scenario there are only one source node sending the packets to Its neighbor and so one.

The Analyze of the results of different Network topologies (line, Mesh and Star) and the same routing protocol (AODV) and the same number of nodes (11) shows that the network performance depends on the network topology that has been used.

4.5 : Scenario 5 - Influence of the number of nodes on the performance : comparing the differente number of nodes for the Mesh topology (11 nodes, 20 nodes, 50 nodes) (Wi-Fi) (AODV)

Tableau 4.7 Scenario 5 Configuration table

Simulation Tool	NS2 Version 2.34
Network dimension	850 X 900
Number of nodes	11 / 25 / 50 nodes
Simulation Time	100 seconds
Maximum queue	50 packets
Data transfer mode	Direct transmission
Maximum air data rates	IEEE 802.11 : 2000 Kbps (Wi-Fi)
Node mobility	Off
Propagation model	Two-ray Ground
Rooting potocols	AODV,
Channel Capacity (Maximum Data rate) = (Video Format Bandwidth)	1 Mb/s (Kilo bit per second (Mbps))
Packet size	1500 Bytes
Transfer Protocol	UDP
Data traffic	CBR
Topology	Mesh

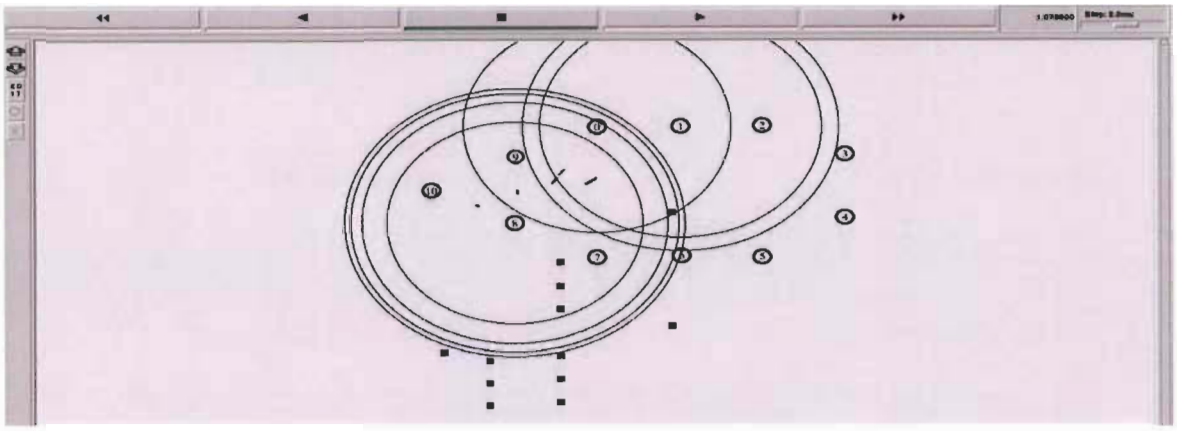


Figure 29 Network Topology Mesh Topology 11 nodes - Our Result

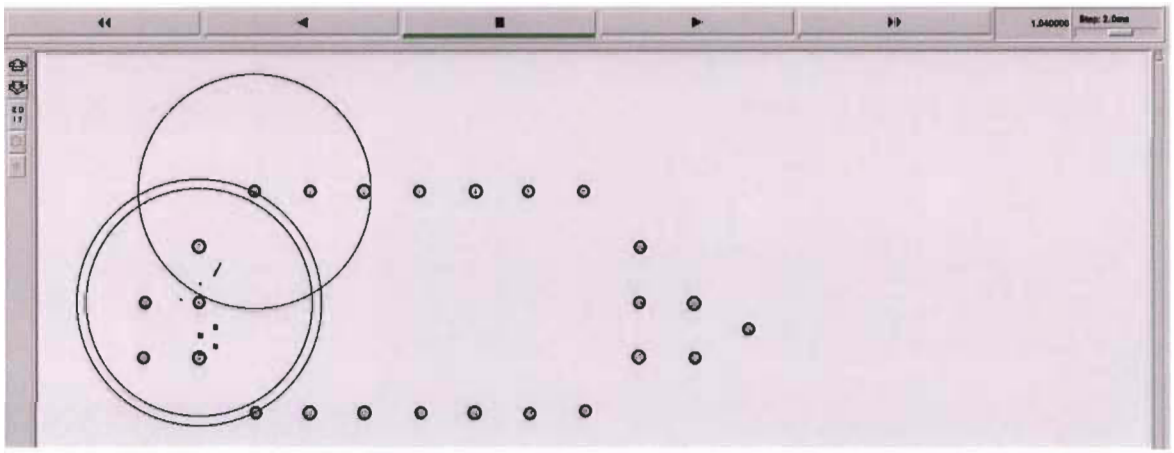


Figure 30 Network Topology Mesh Topology 25 nodes - Our Result

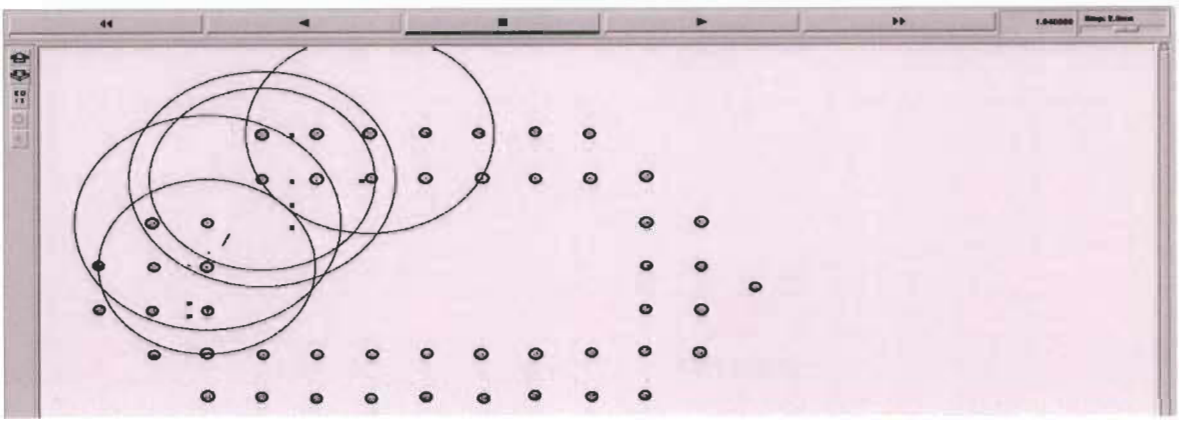


Figure 31 Network Topology Mesh Topology 50 nodes - Our Result

As it is shown in the figure above the topology of the first scenario is composed by 50 nodes representing the smart lights in the street they are spread on a surface of 850 meter X 900 meter. As it is outlined In the table above, three topologies with different number of nodes is going to be tested (11 nodes, 25 nodes, 50 nodes) in the same condition and in the same Network configuration parameters in that, the Simulation Time is going to be 100 seconds, the routing protocols is going to be AODV, the maximum queue allowed 50 packets, the data transfer mode direct transmission, the maximum air data rates (IEEE 802.11: 2000 Kbps (Wi-Fi)) Node mobility will be off and tge Propagation model is Two-ray Ground.

In addition the Packet size will be 1500 Bytes, Transfer Protocol will be the UDP since it is commonly used for video transmission, and for data traffic we are going to utilize the CBR as it is frequently used for streaming multimedia content. And because many nodes are interconnected the topology of our network is going to be the Mesh topology.

4.5.1 Throughput : Influence of the number of nodes on the performance (11 nodes, 20 nodes, 50 nodes) (Wi-Fi) (AODV)

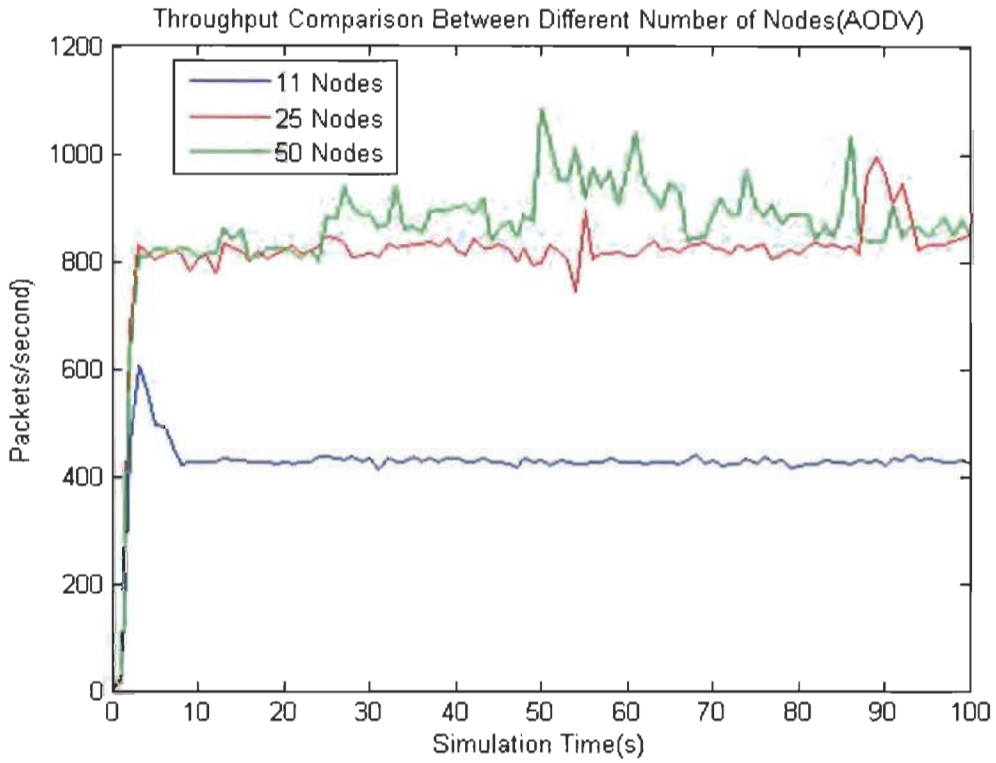


Figure 32 Throughput : Influence of the number of nodes on the performance (Wi-Fi) (AODV). - Our Result

Analyzing Throughput is very popular among network parameters, to find out how the network is performing. Not like in the case of transmitting text messages, video application might require much more bandwidth.

The throughput is a vital performance parameter, in the figure above we compared the throughput of different number of nodes 11, 25 and 50 nodes.

The setup for the simulation scenario functioning at 1.5 Mbps (Data rate) for the three senarios 11 nodes as well as 25 and 50 nodes.

As it is shown in the figure above, we notice that the highest throughput when 11 nodes are employed is 600 packet per second and after 10 seconds of the simulation, the throughput remained steady at around 400 Packet per second until the end of the simulation.

In the beginning of the simulation, the throughput increased dramatically between 0 and 4 seconds until it reached about 600 Packets per seconds, then between the fourth and the tenth seconds it began decreasing steadily until it reached the 400 packets per seconds, that fluctuation in the throughput is due to the rises and declines in collusion and traffic at specific time.

When there is a big traffic, the collision rate escalates and this is how the throughput of the system is affected.

In the second scenario when we deployed 25 nodes, the throughput rose significantly comparing it to the 11 nodes scenario's results.

We observe that the highest throughput when 25 nodes are employed is around 1000 packets per second at the nineteenth second and during most of the time of

the simulation, the throughput remained steady at around 800 Packet per second until the end of the simulation.

In the beginning of the simulation, the throughput increased dramatically between 0 and 4 seconds until it reached about 800 Packets per seconds, and it remained stable, but between the fifteenth and the sixteenth seconds it fluctuated and it has two peaks 700 packets per seconds and 900 packets per seconds. The expansion of the number of nodes rose the throughput of the network.

4.5.2 Jitter : Influence of the number of nodes on the performance (11 nodes, 20 nodes, 50 nodes) (Wi-Fi) (AODV)

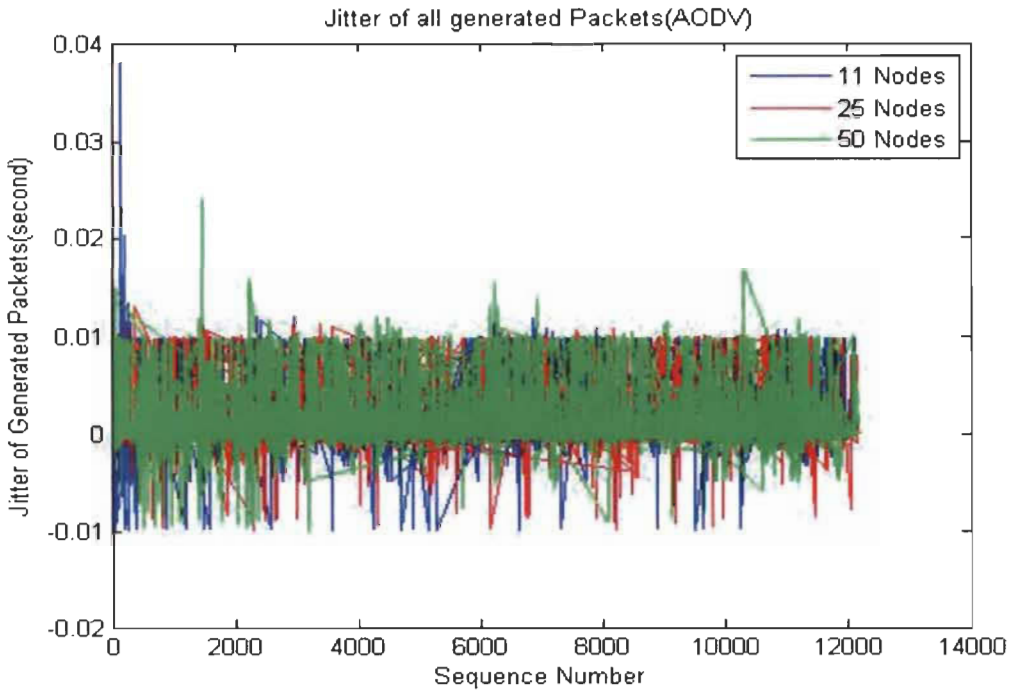


Figure 33 Jitter : Influence of the number of nodes on the performance (Wi-Fi) (AODV). - Our Result

The Jitter consists in the difference in the delay of received packets.

From a sending point of view, packets are transmitted in a steady flow and there is a time frame between the different packets.

Various factors such as network congestion, inappropriate queuing, might cause some issues like the instability of the network data flow, or the fluctuating variation of the delay between every packet.

This figure above shows the collective spreading of the jitter throughout the simulation time for numerous number of nodes (11, 25 and 50 nodes) using the AODV routing protocol.

We notice that in the scenario of the 11 nodes, at the beginning of the simulation, the jitter rose drastically, then it steadied to become similar to the other two scenarios (25 and 50 nodes).

4.5.3 End to End Delay : Influence of the number of nodes on the performance (11 nodes, 20 nodes, 50 nodes) (Wi-Fi) (AODV)

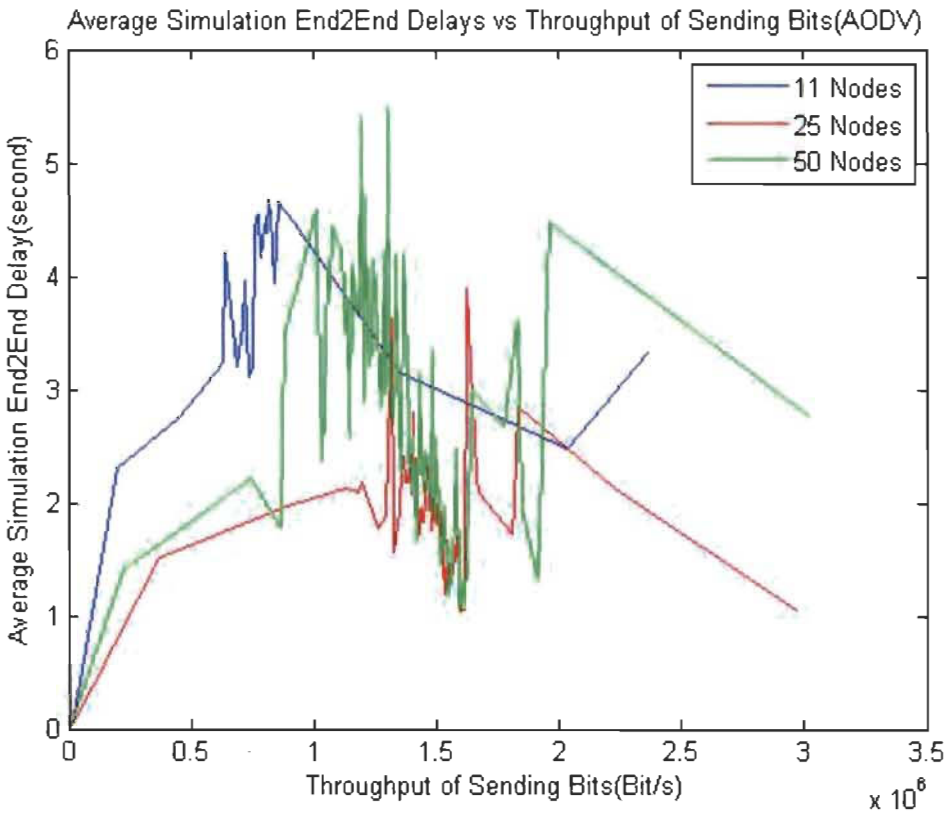


Figure 34 End to End Delay : Influence of the number of nodes on the performance (Wi-Fi) (AODV). - Our Result

The end-to-end delay is the time duration for a packet to be sent plus the time duration to obtain an acknowledgment,

In other words, this delay involves the data transmission time between the two spots of signal.

In the figure above we notice that the overall end to end delay of the 25 nodes

scenario is better when we compare it to the 11 and 50 nodes scenario. And that is due to the congestion of the networks, since in the 11 nodes scenario all the nodes are communicating directly with the source node, however in the 25 nodes scenario there is only 10 nodes communicating directly with the source node.

The Analyze of the results of different routing protocols AODV, DSDV, DSR and different topologies and number of nodes shows that the network performance depends on the routing protocol that has been used, the network topologies, and the number of nodes utilized.

Chapter 5 : Conclusion

In conclusion, the aim of this M.Sc.A. Project was to design and simulate different WSN topologies for the transmission of video data in the context of intelligent streetlights. A routing topology was proposed to achieve the required throughput and to meet specific deployment constraints. Network simulations were carried out to validate the theoretical study.

Furthermore this final project report contained four parts, initially, the introductory chapter, then the literature review chapter which highlighted the different significant points that were required to highlight the theoretical part for required for this final project, this chapter gave a theoretical aspect of the gave the theoretical point of view of the need behind the Wireless Sensor Network as well as some theory behind diverse WSN technologies, especially the routing protocols.

Afterward, in the findings and discussion chapter the main relative findings of the research was underscored; then, from that discussion some conclusion was made such as whether or not the number of nodes utilised in the WSN affects the network performance, And what routing protocol is performing better and so on and so forth, also, this chapter enclosed some figures and tables which were presented to show the result of the testing and simlation steps. And finally, in the conclusion chapter, a summarise of what was performed throughout this report will be presented; then, some recommendations which might improve the

performance of the WSN, was given, so that, it probably will help to achieve further research in the domain of WSN.

Moreover, in the conclusion chapter, a summary of what was performed throughout this dissertation was given, then some recommendations which may foster the performance of the device, was suggested so that, it might help to perform further research in the field of WSN.

In addition, it might be strongly recommended that, the research in the domain of WSN needs to be pushed further whether by researchers and private research institutes or by governments and public research institutes.

Finally, even though this M.Sc.A. dissertation ends up positively and many WSN Topology and routing protocols has been tested and compared, future work might be focused on designing a new routing protocol that might perform better than those tested in this final project (AODV, DSR, DSDV).

Even though this task might requires a lot of time and effort to modify the C++ source code of the Network Simulator NS2 and to compile it again in order to test the new routing protocol, It might be worthwile for further research in a bigger project such as a Ph.D. thesis.

Bibliography (References)

[1] Xuemin Shen, Yi Pan "Fundamentals of Wireless Sensor Networks Theory and practice" Wiley Series on Wireless Communications and Mobile Computing, 2008.

[2] Kazem Sohraby Daniel Minoli Taieb Znati "Wireless Sensor Networks Technology, Protocols, and Applications" Joh n Wiley & Sons, Inc. 2007.

[3] Robert Faludi "Building Wireless Sensor Networks" O'reilly. 2011.

[4] Michael J. McGrath, Cliodhna Ní Scanaill "Sensor Technologies Healthcare, Wellness, and Environmental Applications" ApressOpen, 2014.

[5] Cisco "Fundamentals of Digital Video" 2008
Accessed on December 2016
Link : http ://www.cisco.com/c/en/us/td/docs/solutions/Enterprise/Video/pktvideoaag.html

[6] Mohamed Mostafa A. Azim, Xiaohong Jiang "Wireless Sensor Multimedia Networks Architectures, Protocols, and Applications" CRC Press, 2016.

[7] Ibrahiem M. M. El Emary, S. Ramakrishnan "Wireless Sensor Networks From Theory to Applications" CRC Press, 2014.

[8] I. F. Akyildiz, T. Melodia, and K. R. Chowdury, "Wireless multimedia sensor networks: A survey," Wireless Communications, IEEE, vol. 14, pp. 32-39, 2007.

[9] Yun Ye, Song Ci, A.K. Katsaggelos, Yanwei Liu, and Yi Qian, "Wireless Video Surveillance: A Survey," Access, IEEE , vol.1, no., pp.646,660, 2013.

[10] A. K. Chaurasia, and A. K. Jagannatham, "Dynamic Parallel TCP For Scalable Video Streaming Over MIMO Wireless Networks" The 6th Joint IFIP Wireless and Mobile Networking Conference, Apr. 2013.

[11] R. Yueqing and X Lixin, "A Study on Topological Characteristics of Wireless Sensor Network Based on Complex Network" International Conference on Computer Application and System Modeling, pp. 486 – 489, Oct. 2010.

[12] D. Marpe, H. Schwarz, T. Wiegand, S. Bobe, B. Bross, P. Helle, T. Hinz, H. Kirchhoffer, H. Lakshman, T. Nguyen, S. Oudin, M. Siekmann, K. Sühring, and M. Winken, "Improved Video Compression Technology and the Emerging High Efficiency Video Coding Standard" IEEE International Conference on Consumer Electronics – Berlin, pp. 52 – 56, Sept. 2011.

[13] Cambridge dictionary, (2012) Research, Cambridge dictionary.
Retrieved from <http://dictionary.cambridge.org/>
Accessed on (17/10/2016).

[14] T. Issariyakul, E. Hossain, "Introduction to Network Simulator NS2" Springer 2009

[15] M. Abdelgadir, R. A. Saeed, A. Babiker, "Mobility Routing Model for Vehicular Ad-

hoc Networks (VANETs), Smart City Scenarios", Vehicular Communications 2017

[16] J. Govindasamy, S. Punniakody "A comparative study of reactive, proactive and hybrid routing protocol in wireless sensor network under wormhole attack", Journal of Electrical Systems and Information Technology 2017

[17] A. Pal a, J. Prakash Singh b, P. Dutta, "Path length prediction in MANET under AODV routing: Comparative analysis of ARIMA and MLP model", Egyptian Informatics Journal (2015)

[18] P. Tyagi a, D. Dembla b, "Performance analysis and implementation of proposed mechanism for detection and prevention of security attacks in routing protocols of vehicular ad-hoc network (VANET) " Egyptian Informatics Journal (2017)

[19] B. Blanco a, F. Liberal b, I. Taboada , "Suitability of ad hoc routing in WNR: Performance evaluation and case studies", Ad hoc Networks 11

[20] S. Kim a, O. Lee b, S. Choi b, S. Lee c, "Comparative analysis of link quality metrics and routing protocols for optimal route construction in wireless mesh networks", Ad Hoc Networks 9 (2011)

[21] L. Enciso Quispe a, L. Mengual Galan b, "Behavior of Ad Hoc routing protocols, analyzed for emergency and rescue scenarios, on a real urban area", Expert Systems with Applications 41 (2014)

[22] A. A. Chavana , D. S. Kuruleb, P. U. Derec, "Performance Analysis of AODV and DSDV Routing Protocol in MANET and Modifications in AODV against Black Hole Attack", 7th International Conference on Communication, Computing and Virtualization 2016

[23] S. Mohapatraa, P.Kanungob, "A Performance analysis of AODV, DSR, OLSR and DSDV Routing Protocols using NS2 Simulator", International Conference on Communication Technology and System Design 2011

[24] A. Gorrieri, G. Ferrari, "Irresponsible AODV routing", Vehicular Communications 2015

[25] S. Panichpapiboon, G. Ferrari, Irresponsible forwarding, in: 8th International Conference on Intelligent Transport System Telecommunication (ITST), Phuket, Thailand, 2008, pp. 311–316.

Appendix

Appendix 1: Scenario 1

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
#set val(ifq)    CMUPriQueue               ;#interface Queue type for DSR
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     50                         ;# number of mobilenodes now 50 (it was 49)
set val(rp)     AODV                       ;# routing protocol (DSDV AODV DSR)
set val(x)      650                        ;# X dimension of topography
set val(y)      650                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)


#Open the NS trace file
set tracefile [open out.tr w]

#use the new trace format
#$ns use-newtrace
  $ns use-newtrace

$ns trace-all $tracefile

#Open the NAM trace file
```

```
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#===================================
#        Nodes Definition
#===================================
#Create 49 nodes
set n0 [$ns node]
$n0 set X_ 0
$n0 set Y_ 425
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 25
$n1 set Y_ 425
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 50
$n2 set Y_ 425
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 75
$n3 set Y_ 425
$n3 set Z_ 0.0
```

```
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 100
$n4 set Y_ 425
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 125
$n5 set Y_ 425
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 150
$n6 set Y_ 425
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 175
$n7 set Y_ 425
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 200
$n8 set Y_ 425
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 225
$n9 set Y_ 425
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 250
$n10 set Y_ 425
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

set n11 [$ns node]
$n11 set X_ 275
$n11 set Y_ 425
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20

set n12 [$ns node]
$n12 set X_ 300
$n12 set Y_ 425
```

```
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 325
$n13 set Y_ 425
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 350
$n14 set Y_ 425
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 375
$n15 set Y_ 425
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 400
$n16 set Y_ 425
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 425
$n17 set Y_ 425
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 450
$n18 set Y_ 425
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 475
$n19 set Y_ 425
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 500
$n20 set Y_ 425
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 525
$n21 set Y_ 425
$n21 set Z_ 0.0
```

```
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 550
$n22 set Y_ 425
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 575
$n23 set Y_ 425
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 600
$n24 set Y_ 425
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20
set n25 [$ns node]
$n25 set X_ 625
$n25 set Y_ 425
$n25 set Z_ 0.0
$ns initial_node_pos $n25 20
set n26 [$ns node]
$n26 set X_ 300
$n26 set Y_ 100
$n26 set Z_ 0.0
$ns initial_node_pos $n26 20
set n27 [$ns node]
$n27 set X_ 300
$n27 set Y_ 125
$n27 set Z_ 0.0
$ns initial_node_pos $n27 20
set n28 [$ns node]
$n28 set X_ 300
$n28 set Y_ 150
$n28 set Z_ 0.0
$ns initial_node_pos $n28 20
set n29 [$ns node]
$n29 set X_ 300
$n29 set Y_ 175
$n29 set Z_ 0.0
$ns initial_node_pos $n29 20
set n30 [$ns node]
$n30 set X_ 300
$n30 set Y_ 200
$n30 set Z_ 0.0
$ns initial_node_pos $n30 20
```

```
set n31 [$ns node]
$n31 set X_ 300
$n31 set Y_ 225
$n31 set Z_ 0.0
$ns initial_node_pos $n31 20
set n32 [$ns node]
$n32 set X_ 300
$n32 set Y_ 250
$n32 set Z_ 0.0
$ns initial_node_pos $n32 20
set n33 [$ns node]
$n33 set X_ 300
$n33 set Y_ 275
$n33 set Z_ 0.0
$ns initial_node_pos $n33 20
set n34 [$ns node]
$n34 set X_ 300
$n34 set Y_ 300
$n34 set Z_ 0.0
$ns initial_node_pos $n34 20
set n35 [$ns node]
$n35 set X_ 300
$n35 set Y_ 325
$n35 set Z_ 0.0
$ns initial_node_pos $n35 20
set n36 [$ns node]
$n36 set X_ 300
$n36 set Y_ 350
$n36 set Z_ 0.0
$ns initial_node_pos $n36 20
set n37 [$ns node]
$n37 set X_ 300
$n37 set Y_ 375
$n37 set Z_ 0.0
$ns initial_node_pos $n37 20
set n38 [$ns node]
$n38 set X_ 300
$n38 set Y_ 400
$n38 set Z_ 0.0
$ns initial_node_pos $n38 20
set n39 [$ns node]
$n39 set X_ 300
$n39 set Y_ 425
$n39 set Z_ 0.0
$ns initial_node_pos $n39 20
set n40 [$ns node]
```

```
$n40 set X_ 300
$n40 set Y_ 450
$n40 set Z_ 0.0
$ns initial_node_pos $n40 20
set n41 [$ns node]
$n41 set X_ 300
$n41 set Y_ 475
$n41 set Z_ 0.0
$ns initial_node_pos $n41 20
set n42 [$ns node]
$n42 set X_ 300
$n42 set Y_ 500
$n42 set Z_ 0.0
$ns initial_node_pos $n42 20
set n43 [$ns node]
$n43 set X_ 300
$n43 set Y_ 525
$n43 set Z_ 0.0
$ns initial_node_pos $n43 20
set n44 [$ns node]
$n44 set X_ 300
$n44 set Y_ 550
$n44 set Z_ 0.0
$ns initial_node_pos $n44 20
set n45 [$ns node]
$n45 set X_ 300
$n45 set Y_ 575
$n45 set Z_ 0.0
$ns initial_node_pos $n45 20
set n46 [$ns node]
$n46 set X_ 300
$n46 set Y_ 600
$n46 set Z_ 0.0
$ns initial_node_pos $n46 20
set n47 [$ns node]
$n47 set X_ 300
$n47 set Y_ 625
$n47 set Z_ 0.0
$ns initial_node_pos $n47 20
set n48 [$ns node]
$n48 set X_ 300
$n48 set Y_ 650
$n48 set Z_ 0.0
$ns initial_node_pos $n48 20
set n49 [$ns node]
$n49 set X_ 300
```

```
$n49 set Y_ 675
$n49 set Z_ 0.0
$ns initial_node_pos $n49 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n32 $udp0
set null9 [new Agent/Null]
$ns attach-agent $n19 $null9
$ns connect $udp0 $null9
$udp0 set packetSize_ 512

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n19 $udp1
set null10 [new Agent/Null]
$ns attach-agent $n32 $null10
$ns connect $udp1 $null10
$udp1 set packetSize_ 512

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set null7 [new Agent/Null]
$ns attach-agent $n5 $null7
$ns connect $udp2 $null7
$udp2 set packetSize_ 512

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n44 $udp3
set null8 [new Agent/Null]
$ns attach-agent $n35 $null8
$ns connect $udp3 $null8
$udp3 set packetSize_ 512

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n35 $udp4
set null11 [new Agent/Null]
$ns attach-agent $n14 $null11
$ns connect $udp4 $null11
$udp4 set packetSize_ 512
```

```
#Setup a UDP connection
set udp5 [new Agent/UDP]
$ns attach-agent $n5 $udp5
set null6 [new Agent/Null]
$ns attach-agent $n0 $null6
$ns connect $udp5 $null6
$udp5 set packetSize_ 512


#==================================
#        Applications Definition
#==================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 512
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 90.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 512
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 70.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 512
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 80.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp5
$cbr3 set packetSize_ 512
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
```

```
$ns at 1.0 "$cbr3 start"
$ns at 95.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 512
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp3
$cbr5 set packetSize_ 512
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 70.0 "$cbr5 stop"


#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 1: Scenario 2

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
#set val(ifq)    CMUPriQueue               ;#interface Queue type for DSR
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     50                         ;# number of mobilenodes now 50 (it was 49)
set val(rp)     DSDV                       ;# routing protocol (DSDV AODV DSR)
set val(x)      650                        ;# X dimension of topography
set val(y)      650                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)



#Open the NS trace file
set tracefile [open out.tr w]

#use the new trace format
# $ns use-newtrace

$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
```

```
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#===================================
#        Nodes Definition
#===================================
#Create 49 nodes
set n0 [$ns node]
$n0 set X_ 0
$n0 set Y_ 425
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 25
$n1 set Y_ 425
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 50
$n2 set Y_ 425
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 75
$n3 set Y_ 425
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
```

```
set n4 [$ns node]
$n4 set X_ 100
$n4 set Y_ 425
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 125
$n5 set Y_ 425
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 150
$n6 set Y_ 425
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 175
$n7 set Y_ 425
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 200
$n8 set Y_ 425
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 225
$n9 set Y_ 425
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 250
$n10 set Y_ 425
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

set n11 [$ns node]
$n11 set X_ 275
$n11 set Y_ 425
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20

set n12 [$ns node]
$n12 set X_ 300
$n12 set Y_ 425
$n12 set Z_ 0.0
```

```
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 325
$n13 set Y_ 425
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 350
$n14 set Y_ 425
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 375
$n15 set Y_ 425
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 400
$n16 set Y_ 425
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 425
$n17 set Y_ 425
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 450
$n18 set Y_ 425
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 475
$n19 set Y_ 425
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 500
$n20 set Y_ 425
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 525
$n21 set Y_ 425
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
```

```
set n22 [$ns node]
$n22 set X_ 550
$n22 set Y_ 425
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 575
$n23 set Y_ 425
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 600
$n24 set Y_ 425
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20
set n25 [$ns node]
$n25 set X_ 625
$n25 set Y_ 425
$n25 set Z_ 0.0
$ns initial_node_pos $n25 20
set n26 [$ns node]
$n26 set X_ 300
$n26 set Y_ 100
$n26 set Z_ 0.0
$ns initial_node_pos $n26 20
set n27 [$ns node]
$n27 set X_ 300
$n27 set Y_ 125
$n27 set Z_ 0.0
$ns initial_node_pos $n27 20
set n28 [$ns node]
$n28 set X_ 300
$n28 set Y_ 150
$n28 set Z_ 0.0
$ns initial_node_pos $n28 20
set n29 [$ns node]
$n29 set X_ 300
$n29 set Y_ 175
$n29 set Z_ 0.0
$ns initial_node_pos $n29 20
set n30 [$ns node]
$n30 set X_ 300
$n30 set Y_ 200
$n30 set Z_ 0.0
$ns initial_node_pos $n30 20
set n31 [$ns node]
```

```
$n31 set X_ 300
$n31 set Y_ 225
$n31 set Z_ 0.0
$ns initial_node_pos $n31 20
set n32 [$ns node]
$n32 set X_ 300
$n32 set Y_ 250
$n32 set Z_ 0.0
$ns initial_node_pos $n32 20
set n33 [$ns node]
$n33 set X_ 300
$n33 set Y_ 275
$n33 set Z_ 0.0
$ns initial_node_pos $n33 20
set n34 [$ns node]
$n34 set X_ 300
$n34 set Y_ 300
$n34 set Z_ 0.0
$ns initial_node_pos $n34 20
set n35 [$ns node]
$n35 set X_ 300
$n35 set Y_ 325
$n35 set Z_ 0.0
$ns initial_node_pos $n35 20
set n36 [$ns node]
$n36 set X_ 300
$n36 set Y_ 350
$n36 set Z_ 0.0
$ns initial_node_pos $n36 20
set n37 [$ns node]
$n37 set X_ 300
$n37 set Y_ 375
$n37 set Z_ 0.0
$ns initial_node_pos $n37 20
set n38 [$ns node]
$n38 set X_ 300
$n38 set Y_ 400
$n38 set Z_ 0.0
$ns initial_node_pos $n38 20
set n39 [$ns node]
$n39 set X_ 300
$n39 set Y_ 425
$n39 set Z_ 0.0
$ns initial_node_pos $n39 20
set n40 [$ns node]
$n40 set X_ 300
```

```
$n40 set Y_ 450
$n40 set Z_ 0.0
$ns initial_node_pos $n40 20
set n41 [$ns node]
$n41 set X_ 300
$n41 set Y_ 475
$n41 set Z_ 0.0
$ns initial_node_pos $n41 20
set n42 [$ns node]
$n42 set X_ 300
$n42 set Y_ 500
$n42 set Z_ 0.0
$ns initial_node_pos $n42 20
set n43 [$ns node]
$n43 set X_ 300
$n43 set Y_ 525
$n43 set Z_ 0.0
$ns initial_node_pos $n43 20
set n44 [$ns node]
$n44 set X_ 300
$n44 set Y_ 550
$n44 set Z_ 0.0
$ns initial_node_pos $n44 20
set n45 [$ns node]
$n45 set X_ 300
$n45 set Y_ 575
$n45 set Z_ 0.0
$ns initial_node_pos $n45 20
set n46 [$ns node]
$n46 set X_ 300
$n46 set Y_ 600
$n46 set Z_ 0.0
$ns initial_node_pos $n46 20
set n47 [$ns node]
$n47 set X_ 300
$n47 set Y_ 625
$n47 set Z_ 0.0
$ns initial_node_pos $n47 20
set n48 [$ns node]
$n48 set X_ 300
$n48 set Y_ 650
$n48 set Z_ 0.0
$ns initial_node_pos $n48 20
set n49 [$ns node]
$n49 set X_ 300
$n49 set Y_ 675
```

```
$n49 set Z_ 0.0
$ns initial_node_pos $n49 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n32 $udp0
set null9 [new Agent/Null]
$ns attach-agent $n19 $null9
$ns connect $udp0 $null9
$udp0 set packetSize_ 512

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n19 $udp1
set null10 [new Agent/Null]
$ns attach-agent $n32 $null10
$ns connect $udp1 $null10
$udp1 set packetSize_ 512

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set null7 [new Agent/Null]
$ns attach-agent $n5 $null7
$ns connect $udp2 $null7
$udp2 set packetSize_ 512

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n44 $udp3
set null8 [new Agent/Null]
$ns attach-agent $n35 $null8
$ns connect $udp3 $null8
$udp3 set packetSize_ 512

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n35 $udp4
set null11 [new Agent/Null]
$ns attach-agent $n14 $null11
$ns connect $udp4 $null11
$udp4 set packetSize_ 512
```

```
#Setup a UDP connection
set udp5 [new Agent/UDP]
$ns attach-agent $n5 $udp5
set null6 [new Agent/Null]
$ns attach-agent $n0 $null6
$ns connect $udp5 $null6
$udp5 set packetSize_ 512


#==================================
#        Applications Definition
#==================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 512
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 90.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 512
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 70.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 512
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 80.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp5
$cbr3 set packetSize_ 512
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
```

```
$ns at 95.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 512
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp3
$cbr5 set packetSize_ 512
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 70.0 "$cbr5 stop"


#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 3: Scenario 3 / TCL script 1 (Wi-Fi)

A- Wi-Fi tcl Script

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
#set val(ifq)   CMUPriQueue                ;#interface Queue type for DSR
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     50                         ;# number of mobilenodes now 50 (it was 49)
set val(rp)     AODV                       ;# routing protocol (DSDV AODV DSR)
set val(x)      650                        ;# X dimension of topography
set val(y)      650                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)



#Open the NS trace file
set tracefile [open out.tr w]

#use the new trace format
# $ns use-newtrace

$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
```

```
set chan [new $val(chan)];#Create wireless channel

#================================
#     Mobile node parameter setup
#================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#================================
#       Nodes Definition
#================================
#Create 49 nodes
set n0 [$ns node]
$n0 set X_ 0
$n0 set Y_ 425
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 25
$n1 set Y_ 425
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 50
$n2 set Y_ 425
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 75
$n3 set Y_ 425
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 100
```

```
$n4 set Y_ 425
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 125
$n5 set Y_ 425
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 150
$n6 set Y_ 425
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 175
$n7 set Y_ 425
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 200
$n8 set Y_ 425
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 225
$n9 set Y_ 425
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 250
$n10 set Y_ 425
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

set n11 [$ns node]
$n11 set X_ 275
$n11 set Y_ 425
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20

set n12 [$ns node]
$n12 set X_ 300
$n12 set Y_ 425
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
```

```
$n13 set X_ 325
$n13 set Y_ 425
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 350
$n14 set Y_ 425
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 375
$n15 set Y_ 425
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 400
$n16 set Y_ 425
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 425
$n17 set Y_ 425
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 450
$n18 set Y_ 425
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 475
$n19 set Y_ 425
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 500
$n20 set Y_ 425
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 525
$n21 set Y_ 425
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 550
```

```
$n22 set Y_ 425
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 575
$n23 set Y_ 425
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 600
$n24 set Y_ 425
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20
set n25 [$ns node]
$n25 set X_ 625
$n25 set Y_ 425
$n25 set Z_ 0.0
$ns initial_node_pos $n25 20
set n26 [$ns node]
$n26 set X_ 300
$n26 set Y_ 100
$n26 set Z_ 0.0
$ns initial_node_pos $n26 20
set n27 [$ns node]
$n27 set X_ 300
$n27 set Y_ 125
$n27 set Z_ 0.0
$ns initial_node_pos $n27 20
set n28 [$ns node]
$n28 set X_ 300
$n28 set Y_ 150
$n28 set Z_ 0.0
$ns initial_node_pos $n28 20
set n29 [$ns node]
$n29 set X_ 300
$n29 set Y_ 175
$n29 set Z_ 0.0
$ns initial_node_pos $n29 20
set n30 [$ns node]
$n30 set X_ 300
$n30 set Y_ 200
$n30 set Z_ 0.0
$ns initial_node_pos $n30 20
set n31 [$ns node]
$n31 set X_ 300
$n31 set Y_ 225
```

```
$n31 set Z_ 0.0
$ns initial_node_pos $n31 20
set n32 [$ns node]
$n32 set X_ 300
$n32 set Y_ 250
$n32 set Z_ 0.0
$ns initial_node_pos $n32 20
set n33 [$ns node]
$n33 set X_ 300
$n33 set Y_ 275
$n33 set Z_ 0.0
$ns initial_node_pos $n33 20
set n34 [$ns node]
$n34 set X_ 300
$n34 set Y_ 300
$n34 set Z_ 0.0
$ns initial_node_pos $n34 20
set n35 [$ns node]
$n35 set X_ 300
$n35 set Y_ 325
$n35 set Z_ 0.0
$ns initial_node_pos $n35 20
set n36 [$ns node]
$n36 set X_ 300
$n36 set Y_ 350
$n36 set Z_ 0.0
$ns initial_node_pos $n36 20
set n37 [$ns node]
$n37 set X_ 300
$n37 set Y_ 375
$n37 set Z_ 0.0
$ns initial_node_pos $n37 20
set n38 [$ns node]
$n38 set X_ 300
$n38 set Y_ 400
$n38 set Z_ 0.0
$ns initial_node_pos $n38 20
set n39 [$ns node]
$n39 set X_ 300
$n39 set Y_ 425
$n39 set Z_ 0.0
$ns initial_node_pos $n39 20
set n40 [$ns node]
$n40 set X_ 300
$n40 set Y_ 450
$n40 set Z_ 0.0
```

```
$ns initial_node_pos $n40 20
set n41 [$ns node]
$n41 set X_ 300
$n41 set Y_ 475
$n41 set Z_ 0.0
$ns initial_node_pos $n41 20
set n42 [$ns node]
$n42 set X_ 300
$n42 set Y_ 500
$n42 set Z_ 0.0
$ns initial_node_pos $n42 20
set n43 [$ns node]
$n43 set X_ 300
$n43 set Y_ 525
$n43 set Z_ 0.0
$ns initial_node_pos $n43 20
set n44 [$ns node]
$n44 set X_ 300
$n44 set Y_ 550
$n44 set Z_ 0.0
$ns initial_node_pos $n44 20
set n45 [$ns node]
$n45 set X_ 300
$n45 set Y_ 575
$n45 set Z_ 0.0
$ns initial_node_pos $n45 20
set n46 [$ns node]
$n46 set X_ 300
$n46 set Y_ 600
$n46 set Z_ 0.0
$ns initial_node_pos $n46 20
set n47 [$ns node]
$n47 set X_ 300
$n47 set Y_ 625
$n47 set Z_ 0.0
$ns initial_node_pos $n47 20
set n48 [$ns node]
$n48 set X_ 300
$n48 set Y_ 650
$n48 set Z_ 0.0
$ns initial_node_pos $n48 20
set n49 [$ns node]
$n49 set X_ 300
$n49 set Y_ 675
$n49 set Z_ 0.0
$ns initial_node_pos $n49 20
```

```
#==================================
#        Agents Definition
#==================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n32 $udp0
set null9 [new Agent/Null]
$ns attach-agent $n19 $null9
$ns connect $udp0 $null9
$udp0 set packetSize_ 512

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n19 $udp1
set null10 [new Agent/Null]
$ns attach-agent $n32 $null10
$ns connect $udp1 $null10
$udp1 set packetSize_ 512

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set null7 [new Agent/Null]
$ns attach-agent $n5 $null7
$ns connect $udp2 $null7
$udp2 set packetSize_ 512

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n44 $udp3
set null8 [new Agent/Null]
$ns attach-agent $n35 $null8
$ns connect $udp3 $null8
$udp3 set packetSize_ 512

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n35 $udp4
set null11 [new Agent/Null]
$ns attach-agent $n14 $null11
$ns connect $udp4 $null11
$udp4 set packetSize_ 512

#Setup a UDP connection
set udp5 [new Agent/UDP]
```

```
$ns attach-agent $n5 $udp5
set null6 [new Agent/Null]
$ns attach-agent $n0 $null6
$ns connect $udp5 $null6
$udp5 set packetSize_ 512


#==================================
#       Applications Definition
#==================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 512
$cbr0 set rate_ 260Kb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 90.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 512
$cbr1 set rate_ 260Kb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 70.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 512
$cbr2 set rate_ 260Kb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 80.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp5
$cbr3 set packetSize_ 512
$cbr3 set rate_ 260Kb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
$ns at 95.0 "$cbr3 stop"
```

```
#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 512
$cbr4 set rate_ 260Kb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp3
$cbr5 set packetSize_ 512
$cbr5 set rate_ 260Kb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 70.0 "$cbr5 stop"


#====================================
#        Termination
#====================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 3: Scenario 3 / TCL script 2 (Zigbee)

B- Zigbee tcl Script

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
#set val(prop)  Propagation/TwoRayGround   ;# radio-propagation model

set val(prop)   Propagation/Shadowing      ;# Propragation Model: Shadowing
(Shadowing/TwoRayGround/FreeSpace)

set val(netif)  Phy/WirelessPhy/802_15_4   ;# network interface type
set val(mac)    Mac/802_15_4               ;# MAC type

set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
#set val(ifq)    CMUPriQueue               ;#interface Queue type for DSR
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     50                         ;# number of mobilenodes now 50 (it was 49)
set val(rp)     AODV                       ;# routing protocol (DSDV AODV DSR)
set val(x)      650                        ;# X dimension of topography
set val(y)      650                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)


#Open the NS trace file
set tracefile [open out.tr w]

#use the new trace format
# $ns use-newtrace

$ns trace-all $tracefile

#Open the NAM trace file
```

```
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#===================================
#        Nodes Definition
#===================================
#Create 49 nodes
set n0 [$ns node]
$n0 set X_ 0
$n0 set Y_ 425
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 25
$n1 set Y_ 425
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 50
$n2 set Y_ 425
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 75
$n3 set Y_ 425
$n3 set Z_ 0.0
```

```
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 100
$n4 set Y_ 425
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 125
$n5 set Y_ 425
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 150
$n6 set Y_ 425
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 175
$n7 set Y_ 425
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 200
$n8 set Y_ 425
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 225
$n9 set Y_ 425
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 250
$n10 set Y_ 425
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

set n11 [$ns node]
$n11 set X_ 275
$n11 set Y_ 425
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20

set n12 [$ns node]
$n12 set X_ 300
$n12 set Y_ 425
```

```
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 325
$n13 set Y_ 425
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 350
$n14 set Y_ 425
$n14 set Z_ 0.0
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 375
$n15 set Y_ 425
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 400
$n16 set Y_ 425
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 425
$n17 set Y_ 425
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 450
$n18 set Y_ 425
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 475
$n19 set Y_ 425
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 500
$n20 set Y_ 425
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 525
$n21 set Y_ 425
$n21 set Z_ 0.0
```

```
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 550
$n22 set Y_ 425
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 575
$n23 set Y_ 425
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
set n24 [$ns node]
$n24 set X_ 600
$n24 set Y_ 425
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20
set n25 [$ns node]
$n25 set X_ 625
$n25 set Y_ 425
$n25 set Z_ 0.0
$ns initial_node_pos $n25 20
set n26 [$ns node]
$n26 set X_ 300
$n26 set Y_ 100
$n26 set Z_ 0.0
$ns initial_node_pos $n26 20
set n27 [$ns node]
$n27 set X_ 300
$n27 set Y_ 125
$n27 set Z_ 0.0
$ns initial_node_pos $n27 20
set n28 [$ns node]
$n28 set X_ 300
$n28 set Y_ 150
$n28 set Z_ 0.0
$ns initial_node_pos $n28 20
set n29 [$ns node]
$n29 set X_ 300
$n29 set Y_ 175
$n29 set Z_ 0.0
$ns initial_node_pos $n29 20
set n30 [$ns node]
$n30 set X_ 300
$n30 set Y_ 200
$n30 set Z_ 0.0
$ns initial_node_pos $n30 20
```

```
set n31 [$ns node]
$n31 set X_ 300
$n31 set Y_ 225
$n31 set Z_ 0.0
$ns initial_node_pos $n31 20
set n32 [$ns node]
$n32 set X_ 300
$n32 set Y_ 250
$n32 set Z_ 0.0
$ns initial_node_pos $n32 20
set n33 [$ns node]
$n33 set X_ 300
$n33 set Y_ 275
$n33 set Z_ 0.0
$ns initial_node_pos $n33 20
set n34 [$ns node]
$n34 set X_ 300
$n34 set Y_ 300
$n34 set Z_ 0.0
$ns initial_node_pos $n34 20
set n35 [$ns node]
$n35 set X_ 300
$n35 set Y_ 325
$n35 set Z_ 0.0
$ns initial_node_pos $n35 20
set n36 [$ns node]
$n36 set X_ 300
$n36 set Y_ 350
$n36 set Z_ 0.0
$ns initial_node_pos $n36 20
set n37 [$ns node]
$n37 set X_ 300
$n37 set Y_ 375
$n37 set Z_ 0.0
$ns initial_node_pos $n37 20
set n38 [$ns node]
$n38 set X_ 300
$n38 set Y_ 400
$n38 set Z_ 0.0
$ns initial_node_pos $n38 20
set n39 [$ns node]
$n39 set X_ 300
$n39 set Y_ 425
$n39 set Z_ 0.0
$ns initial_node_pos $n39 20
set n40 [$ns node]
```

```
$n40 set X_ 300
$n40 set Y_ 450
$n40 set Z_ 0.0
$ns initial_node_pos $n40 20
set n41 [$ns node]
$n41 set X_ 300
$n41 set Y_ 475
$n41 set Z_ 0.0
$ns initial_node_pos $n41 20
set n42 [$ns node]
$n42 set X_ 300
$n42 set Y_ 500
$n42 set Z_ 0.0
$ns initial_node_pos $n42 20
set n43 [$ns node]
$n43 set X_ 300
$n43 set Y_ 525
$n43 set Z_ 0.0
$ns initial_node_pos $n43 20
set n44 [$ns node]
$n44 set X_ 300
$n44 set Y_ 550
$n44 set Z_ 0.0
$ns initial_node_pos $n44 20
set n45 [$ns node]
$n45 set X_ 300
$n45 set Y_ 575
$n45 set Z_ 0.0
$ns initial_node_pos $n45 20
set n46 [$ns node]
$n46 set X_ 300
$n46 set Y_ 600
$n46 set Z_ 0.0
$ns initial_node_pos $n46 20
set n47 [$ns node]
$n47 set X_ 300
$n47 set Y_ 625
$n47 set Z_ 0.0
$ns initial_node_pos $n47 20
set n48 [$ns node]
$n48 set X_ 300
$n48 set Y_ 650
$n48 set Z_ 0.0
$ns initial_node_pos $n48 20
set n49 [$ns node]
$n49 set X_ 300
```

```
$n49 set Y_ 675
$n49 set Z_ 0.0
$ns initial_node_pos $n49 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n32 $udp0
set null9 [new Agent/Null]
$ns attach-agent $n19 $null9
$ns connect $udp0 $null9
$udp0 set packetSize_ 512

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n19 $udp1
set null10 [new Agent/Null]
$ns attach-agent $n32 $null10
$ns connect $udp1 $null10
$udp1 set packetSize_ 512

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set null7 [new Agent/Null]
$ns attach-agent $n5 $null7
$ns connect $udp2 $null7
$udp2 set packetSize_ 512

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n44 $udp3
set null8 [new Agent/Null]
$ns attach-agent $n35 $null8
$ns connect $udp3 $null8
$udp3 set packetSize_ 512

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n35 $udp4
set null11 [new Agent/Null]
$ns attach-agent $n14 $null11
$ns connect $udp4 $null11
$udp4 set packetSize_ 512
```

```
#Setup a UDP connection
set udp5 [new Agent/UDP]
$ns attach-agent $n5 $udp5
set null6 [new Agent/Null]
$ns attach-agent $n0 $null6
$ns connect $udp5 $null6
$udp5 set packetSize_ 512

#==================================
#       Applications Definition
#==================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 512
$cbr0 set rate_ 260Kb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 90.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 512
$cbr1 set rate_ 260Kb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 70.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 512
$cbr2 set rate_ 260Kb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 80.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp5
$cbr3 set packetSize_ 512
$cbr3 set rate_ 260Kb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
```

```
$ns at 95.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 512
$cbr4 set rate_ 260Kb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 80.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp3
$cbr5 set packetSize_ 512
$cbr5 set rate_ 260Kb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 70.0 "$cbr5 stop"


#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 4: Scenario 4 / TCL script 1 (Line Topology)

A Line Topology

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     11                         ;# number of mobilenodes
set val(rp)     DSDV                       ;# routing protocol
set val(x)      1302                       ;# X dimension of topography
set val(y)      502                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
```

```
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#===================================
#        Nodes Definition
#===================================
#Create 11 nodes
set n0 [$ns node]
$n0 set X_ 199
$n0 set Y_ 401
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 299
$n1 set Y_ 401
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 401
$n2 set Y_ 399
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 502
$n3 set Y_ 400
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 602
$n4 set Y_ 402
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
```

```
$n5 set X_ 699
$n5 set Y_ 399
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 799
$n6 set Y_ 397
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 901
$n7 set Y_ 400
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 998
$n8 set Y_ 400
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 1101
$n9 set Y_ 402
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 1202
$n10 set Y_ 400
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set null9 [new Agent/Null]
$ns attach-agent $n10 $null9
$ns connect $udp0 $null9
$udp0 set packetSize_ 1500

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n2 $udp1
set null8 [new Agent/Null]
$ns attach-agent $n10 $null8
```

```
$ns connect $udp1 $null8
$udp1 set packetSize_ 1500

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n4 $udp2
set null7 [new Agent/Null]
$ns attach-agent $n10 $null7
$ns connect $udp2 $null7
$udp2 set packetSize_ 1500

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n6 $udp3
set null6 [new Agent/Null]
$ns attach-agent $n10 $null6
$ns connect $udp3 $null6
$udp3 set packetSize_ 1500

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n8 $udp4
set null5 [new Agent/Null]
$ns attach-agent $n10 $null5
$ns connect $udp4 $null5
$udp4 set packetSize_ 1500


#=================================
#       Applications Definition
#=================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 99.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
```

```
$ns at 1.0 "$cbr1 start"
$ns at 99.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 99.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
$ns at 99.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 99.0 "$cbr4 stop"


#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
```

```
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 4: Scenario 4 / TCL script 2 (Mesh Topology)

B Mesh Topology

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     11                         ;# number of mobilenodes
set val(rp)     DSDV                       ;# routing protocol
set val(x)      801                        ;# X dimension of topography
set val(y)      600                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel
```

```
#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#===================================
#        Nodes Definition
#===================================
#Create 11 nodes
set n0 [$ns node]
$n0 set X_ 401
$n0 set Y_ 498
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 502
$n1 set Y_ 499
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 601
$n2 set Y_ 500
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 701
$n3 set Y_ 457
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 701
$n4 set Y_ 360
```

```
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 601
$n5 set Y_ 300
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 503
$n6 set Y_ 301
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 401
$n7 set Y_ 297
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 301
$n8 set Y_ 351
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 302
$n9 set Y_ 452
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 200
$n10 set Y_ 400
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

#====================================
#        Agents Definition
#====================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n10 $udp0
set null10 [new Agent/Null]
$ns attach-agent $n8 $null10
$ns connect $udp0 $null10
$udp0 set packetSize_ 1500

#Setup a UDP connection
set udp1 [new Agent/UDP]
```

```
$ns attach-agent $n9 $udp1
set null12 [new Agent/Null]
$ns attach-agent $n8 $null12
$ns connect $udp1 $null12
$udp1 set packetSize_ 1500

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set null13 [new Agent/Null]
$ns attach-agent $n8 $null13
$ns connect $udp2 $null13
$udp2 set packetSize_ 1500

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n1 $udp3
set null14 [new Agent/Null]
$ns attach-agent $n8 $null14
$ns connect $udp3 $null14
$udp3 set packetSize_ 1500

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n2 $udp4
set null16 [new Agent/Null]
$ns attach-agent $n5 $null16
$ns connect $udp4 $null16
$udp4 set packetSize_ 1500

#Setup a UDP connection
set udp5 [new Agent/UDP]
$ns attach-agent $n3 $udp5
set null15 [new Agent/Null]
$ns attach-agent $n5 $null15
$ns connect $udp5 $null15
$udp5 set packetSize_ 1500

#Setup a UDP connection
set udp6 [new Agent/UDP]
$ns attach-agent $n4 $udp6
set null11 [new Agent/Null]
$ns attach-agent $n5 $null11
$ns connect $udp6 $null11
$udp6 set packetSize_ 1500
```

```
#Setup a UDP connection
set udp8 [new Agent/UDP]
$ns attach-agent $n6 $udp8
set null17 [new Agent/Null]
$ns attach-agent $n5 $null17
$ns connect $udp8 $null17
$udp8 set packetSize_ 1500

#Setup a UDP connection
set udp9 [new Agent/UDP]
$ns attach-agent $n7 $udp9
set null18 [new Agent/Null]
$ns attach-agent $n5 $null18
$ns connect $udp9 $null18
$udp9 set packetSize_ 1500


#==================================
#       Applications Definition
#==================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 99.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 99.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 99.0 "$cbr2 stop"
```

```
#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
$ns at 99.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 99.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 99.0 "$cbr5 stop"

#Setup a CBR Application over UDP connection
set cbr6 [new Application/Traffic/CBR]
$cbr6 attach-agent $udp6
$cbr6 set packetSize_ 1000
$cbr6 set rate_ 1.0Mb
$cbr6 set random_ null
$ns at 1.0 "$cbr6 start"
$ns at 99.0 "$cbr6 stop"

#Setup a CBR Application over UDP connection
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp8
$cbr7 set packetSize_ 1000
$cbr7 set rate_ 1.0Mb
$cbr7 set random_ null
$ns at 1.0 "$cbr7 start"
$ns at 99.0 "$cbr7 stop"
```

```
#Setup a CBR Application over UDP connection
set cbr8 [new Application/Traffic/CBR]
$cbr8 attach-agent $udp9
$cbr8 set packetSize_ 1000
$cbr8 set rate_ 1.0Mb
$cbr8 set random_ null
$ns at 1.0 "$cbr8 start"
$ns at 99.0 "$cbr8 stop"


#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 4: Scenario 4 / TCL script 3 (Star Topology)

```
C Star Topology
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     11                         ;# number of mobilenodes
set val(rp)     DSDV                       ;# routing protocol
set val(x)      900                        ;# X dimension of topography
set val(y)      603                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel
```

```
#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#===================================
#        Nodes Definition
#===================================
#Create 11 nodes
set n0 [$ns node]
$n0 set X_ 601
$n0 set Y_ 399
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 702
$n1 set Y_ 500
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 600
$n2 set Y_ 503
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 499
$n3 set Y_ 502
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 410
$n4 set Y_ 364
```

```
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 501
$n5 set Y_ 303
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 602
$n6 set Y_ 304
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 701
$n7 set Y_ 297
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 799
$n8 set Y_ 358
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 800
$n9 set Y_ 448
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 410
$n10 set Y_ 446
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n1 $udp4
set null13 [new Agent/Null]
$ns attach-agent $n0 $null13
$ns connect $udp4 $null13
$udp4 set packetSize_ 1500

#Setup a UDP connection
set udp5 [new Agent/UDP]
```

```
$ns attach-agent $n2 $udp5
set null15 [new Agent/Null]
$ns attach-agent $n0 $null15
$ns connect $udp5 $null15
$udp5 set packetSize_ 1500

#Setup a UDP connection
set udp6 [new Agent/UDP]
$ns attach-agent $n3 $udp6
set null22 [new Agent/Null]
$ns attach-agent $n0 $null22
$ns connect $udp6 $null22
$udp6 set packetSize_ 1500

#Setup a UDP connection
set udp7 [new Agent/UDP]
$ns attach-agent $n4 $udp7
set null21 [new Agent/Null]
$ns attach-agent $n0 $null21
$ns connect $udp7 $null21
$udp7 set packetSize_ 1500

#Setup a UDP connection
set udp8 [new Agent/UDP]
$ns attach-agent $n5 $udp8
set null20 [new Agent/Null]
$ns attach-agent $n0 $null20
$ns connect $udp8 $null20
$udp8 set packetSize_ 1500

#Setup a UDP connection
set udp9 [new Agent/UDP]
$ns attach-agent $n6 $udp9
set null19 [new Agent/Null]
$ns attach-agent $n0 $null19
$ns connect $udp9 $null19
$udp9 set packetSize_ 1500

#Setup a UDP connection
set udp10 [new Agent/UDP]
$ns attach-agent $n7 $udp10
set null18 [new Agent/Null]
$ns attach-agent $n0 $null18
$ns connect $udp10 $null18
$udp10 set packetSize_ 1500
```

```
#Setup a UDP connection
set udp11 [new Agent/UDP]
$ns attach-agent $n8 $udp11
set null17 [new Agent/Null]
$ns attach-agent $n0 $null17
$ns connect $udp11 $null17
$udp11 set packetSize_ 1500

#Setup a UDP connection
set udp12 [new Agent/UDP]
$ns attach-agent $n10 $udp12
set null23 [new Agent/Null]
$ns attach-agent $n0 $null23
$ns connect $udp12 $null23
$udp12 set packetSize_ 1500

#Setup a UDP connection
set udp14 [new Agent/UDP]
$ns attach-agent $n9 $udp14
set null16 [new Agent/Null]
$ns attach-agent $n0 $null16
$ns connect $udp14 $null16
$udp14 set packetSize_ 1500


#=================================
#        Applications Definition
#=================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp4
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 99.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp5
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 99.0 "$cbr1 stop"
```

```
#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp6
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 99.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp12
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
$ns at 99.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp7
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 99.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp8
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 99.0 "$cbr5 stop"

#Setup a CBR Application over UDP connection
set cbr6 [new Application/Traffic/CBR]
$cbr6 attach-agent $udp9
$cbr6 set packetSize_ 1000
$cbr6 set rate_ 1.0Mb
$cbr6 set random_ null
$ns at 1.0 "$cbr6 start"
$ns at 99.0 "$cbr6 stop"

#Setup a CBR Application over UDP connection
```

```
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp10
$cbr7 set packetSize_ 1000
$cbr7 set rate_ 1.0Mb
$cbr7 set random_ null
$ns at 1.0 "$cbr7 start"
$ns at 99.0 "$cbr7 stop"

#Setup a CBR Application over UDP connection
set cbr8 [new Application/Traffic/CBR]
$cbr8 attach-agent $udp11
$cbr8 set packetSize_ 1000
$cbr8 set rate_ 1.0Mb
$cbr8 set random_ null
$ns at 1.0 "$cbr8 start"
$ns at 99.0 "$cbr8 stop"

#Setup a CBR Application over UDP connection
set cbr9 [new Application/Traffic/CBR]
$cbr9 attach-agent $udp14
$cbr9 set packetSize_ 1000
$cbr9 set rate_ 1.0Mb
$cbr9 set random_ null
$ns at 1.0 "$cbr9 start"
$ns at 99.0 "$cbr9 stop"


#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 5: Scenario 5 / TCL script 1 (Mesh Topology 11 nodes)

A. Mesh Topology 11 nodes

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     11                         ;# number of mobilenodes
set val(rp)     DSDV                       ;# routing protocol
set val(x)      801                        ;# X dimension of topography
set val(y)      600                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#       Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
```

```
$ns node-config -adhocRouting  $val(rp) \
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#================================
#       Nodes Definition
#================================
#Create 11 nodes
set n0 [$ns node]
$n0 set X_ 401
$n0 set Y_ 498
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 502
$n1 set Y_ 499
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 601
$n2 set Y_ 500
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 701
$n3 set Y_ 457
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 701
$n4 set Y_ 360
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 601
```

```
$n5 set Y_ 300
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 503
$n6 set Y_ 301
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 401
$n7 set Y_ 297
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 301
$n8 set Y_ 351
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 302
$n9 set Y_ 452
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 200
$n10 set Y_ 400
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp0 [new Agent/UDP]
$ns attach-agent $n10 $udp0
set null10 [new Agent/Null]
$ns attach-agent $n8 $null10
$ns connect $udp0 $null10
$udp0 set packetSize_ 1500

#Setup a UDP connection
set udp1 [new Agent/UDP]
$ns attach-agent $n9 $udp1
set null12 [new Agent/Null]
$ns attach-agent $n8 $null12
$ns connect $udp1 $null12
```

```
$udp1 set packetSize_ 1500

#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n0 $udp2
set null13 [new Agent/Null]
$ns attach-agent $n8 $null13
$ns connect $udp2 $null13
$udp2 set packetSize_ 1500

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n1 $udp3
set null14 [new Agent/Null]
$ns attach-agent $n8 $null14
$ns connect $udp3 $null14
$udp3 set packetSize_ 1500

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n2 $udp4
set null16 [new Agent/Null]
$ns attach-agent $n5 $null16
$ns connect $udp4 $null16
$udp4 set packetSize_ 1500

#Setup a UDP connection
set udp5 [new Agent/UDP]
$ns attach-agent $n3 $udp5
set null15 [new Agent/Null]
$ns attach-agent $n5 $null15
$ns connect $udp5 $null15
$udp5 set packetSize_ 1500

#Setup a UDP connection
set udp6 [new Agent/UDP]
$ns attach-agent $n4 $udp6
set null11 [new Agent/Null]
$ns attach-agent $n5 $null11
$ns connect $udp6 $null11
$udp6 set packetSize_ 1500

#Setup a UDP connection
set udp8 [new Agent/UDP]
$ns attach-agent $n6 $udp8
set null17 [new Agent/Null]
```

```
$ns attach-agent $n5 $null17
$ns connect $udp8 $null17
$udp8 set packetSize_ 1500

#Setup a UDP connection
set udp9 [new Agent/UDP]
$ns attach-agent $n7 $udp9
set null18 [new Agent/Null]
$ns attach-agent $n5 $null18
$ns connect $udp9 $null18
$udp9 set packetSize_ 1500


#================================
#       Applications Definition
#================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 99.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp1
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 99.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp2
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 99.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp3
```

```
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
$ns at 99.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp4
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 99.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp5
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 99.0 "$cbr5 stop"

#Setup a CBR Application over UDP connection
set cbr6 [new Application/Traffic/CBR]
$cbr6 attach-agent $udp6
$cbr6 set packetSize_ 1000
$cbr6 set rate_ 1.0Mb
$cbr6 set random_ null
$ns at 1.0 "$cbr6 start"
$ns at 99.0 "$cbr6 stop"

#Setup a CBR Application over UDP connection
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp8
$cbr7 set packetSize_ 1000
$cbr7 set rate_ 1.0Mb
$cbr7 set random_ null
$ns at 1.0 "$cbr7 start"
$ns at 99.0 "$cbr7 stop"

#Setup a CBR Application over UDP connection
set cbr8 [new Application/Traffic/CBR]
$cbr8 attach-agent $udp9
$cbr8 set packetSize_ 1000
```

```
$cbr8 set rate_ 1.0Mb
$cbr8 set random_ null
$ns at 1.0 "$cbr8 start"
$ns at 99.0 "$cbr8 stop"


#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam out.nam &
   exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 5: Scenario 5 / TCL script 2 (Mesh Topology 25 nodes)

B. . Mesh Topology 25 nodes

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     25                         ;# number of mobilenodes
set val(rp)     DSDV                       ;# routing protocol
set val(x)      1839                       ;# X dimension of topography
set val(y)      100                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#       Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
```

```
            -llType        $val(ll) \
            -macType       $val(mac) \
            -ifqType       $val(ifq) \
            -ifqLen        $val(ifqlen) \
            -antType       $val(ant) \
            -propType      $val(prop) \
            -phyType       $val(netif) \
            -channel       $chan \
            -topoInstance  $topo \
            -agentTrace    ON \
            -routerTrace   ON \
            -macTrace      ON \
            -movementTrace ON

#================================
#       Nodes Definition
#================================
#Create 25 nodes
set n0 [$ns node]
$n0 set X_ 300
$n0 set Y_ 499
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 401
$n1 set Y_ 499
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 500
$n2 set Y_ 500
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 600
$n3 set Y_ 500
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 703
$n4 set Y_ 500
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 799
$n5 set Y_ 499
```

```
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 900
$n6 set Y_ 499
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 200
$n7 set Y_ 400
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 199
$n8 set Y_ 300
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 200
$n9 set Y_ 200
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 302
$n10 set Y_ 100
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 400
$n11 set Y_ 100
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20
set n12 [$ns node]
$n12 set X_ 500
$n12 set Y_ 100
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 601
$n13 set Y_ 101
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 700
$n14 set Y_ 100
$n14 set Z_ 0.0
```

```
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 800
$n15 set Y_ 99
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 901
$n16 set Y_ 103
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 1000
$n17 set Y_ 201
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 1001
$n18 set Y_ 299
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 1002
$n19 set Y_ 399
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 1101
$n20 set Y_ 298
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 1102
$n21 set Y_ 200
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 101
$n22 set Y_ 299
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 98
$n23 set Y_ 200
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
```

```
set n24 [$ns node]
$n24 set X_ 1200
$n24 set Y_ 251
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n22 $udp2
set null0 [new Agent/Null]
$ns attach-agent $n8 $null0
$ns connect $udp2 $null0
$udp2 set packetSize_ 1500

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n23 $udp3
set null29 [new Agent/Null]
$ns attach-agent $n8 $null29
$ns connect $udp3 $null29
$udp3 set packetSize_ 1500

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n9 $udp4
set null28 [new Agent/Null]
$ns attach-agent $n8 $null28
$ns connect $udp4 $null28
$udp4 set packetSize_ 1500

#Setup a UDP connection
set udp5 [new Agent/UDP]
$ns attach-agent $n7 $udp5
set null17 [new Agent/Null]
$ns attach-agent $n8 $null17
$ns connect $udp5 $null17
$udp5 set packetSize_ 1500

#Setup a UDP connection
set udp6 [new Agent/UDP]
$ns attach-agent $n0 $udp6
set null18 [new Agent/Null]
$ns attach-agent $n8 $null18
```

```
$ns connect $udp6 $null18
$udp6 set packetSize_ 1500

#Setup a UDP connection
set udp7 [new Agent/UDP]
$ns attach-agent $n1 $udp7
set null19 [new Agent/Null]
$ns attach-agent $n8 $null19
$ns connect $udp7 $null19
$udp7 set packetSize_ 1500

#Setup a UDP connection
set udp8 [new Agent/UDP]
$ns attach-agent $n2 $udp8
set null20 [new Agent/Null]
$ns attach-agent $n8 $null20
$ns connect $udp8 $null20
$udp8 set packetSize_ 1500

#Setup a UDP connection
set udp9 [new Agent/UDP]
$ns attach-agent $n3 $udp9
set null21 [new Agent/Null]
$ns attach-agent $n5 $null21
$ns connect $udp9 $null21
$udp9 set packetSize_ 1500

#Setup a UDP connection
set udp10 [new Agent/UDP]
$ns attach-agent $n4 $udp10
set null22 [new Agent/Null]
$ns attach-agent $n5 $null22
$ns connect $udp10 $null22
$udp10 set packetSize_ 1500

#Setup a UDP connection
set udp11 [new Agent/UDP]
$ns attach-agent $n6 $udp11
set null24 [new Agent/Null]
$ns attach-agent $n5 $null24
$ns connect $udp11 $null24
$udp11 set packetSize_ 1500

#Setup a UDP connection
set udp12 [new Agent/UDP]
$ns attach-agent $n19 $udp12
```

```
set null23 [new Agent/Null]
$ns attach-agent $n5 $null23
$ns connect $udp12 $null23
$udp12 set packetSize_ 1500

#Setup a UDP connection
set udp13 [new Agent/UDP]
$ns attach-agent $n18 $udp13
set null25 [new Agent/Null]
$ns attach-agent $n5 $null25
$ns connect $udp13 $null25
$udp13 set packetSize_ 1500

#Setup a UDP connection
set udp14 [new Agent/UDP]
$ns attach-agent $n15 $udp14
set null26 [new Agent/Null]
$ns attach-agent $n5 $null26
$ns connect $udp14 $null26
$udp14 set packetSize_ 1500

#Setup a UDP connection
set udp15 [new Agent/UDP]
$ns attach-agent $n13 $udp15
set null27 [new Agent/Null]
$ns attach-agent $n5 $null27
$ns connect $udp15 $null27
$udp15 set packetSize_ 1500

#Setup a UDP connection
set udp16 [new Agent/UDP]
$ns attach-agent $n11 $udp16
set null1 [new Agent/Null]
$ns attach-agent $n5 $null1
$ns connect $udp16 $null1
$udp16 set packetSize_ 1500


#===================================
#        Applications Definition
#===================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp2
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
```

```
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 99.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp5
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
$ns at 99.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp6
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 99.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp7
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
$ns at 99.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp8
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 99.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp9
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
```

```
$ns at 1.0 "$cbr5 start"
$ns at 99.0 "$cbr5 stop"

#Setup a CBR Application over UDP connection
set cbr6 [new Application/Traffic/CBR]
$cbr6 attach-agent $udp10
$cbr6 set packetSize_ 1000
$cbr6 set rate_ 1.0Mb
$cbr6 set random_ null
$ns at 1.0 "$cbr6 start"
$ns at 99.0 "$cbr6 stop"

#Setup a CBR Application over UDP connection
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp11
$cbr7 set packetSize_ 1000
$cbr7 set rate_ 1.0Mb
$cbr7 set random_ null
$ns at 1.0 "$cbr7 start"
$ns at 99.0 "$cbr7 stop"

#Setup a CBR Application over UDP connection
set cbr8 [new Application/Traffic/CBR]
$cbr8 attach-agent $udp12
$cbr8 set packetSize_ 1000
$cbr8 set rate_ 1.0Mb
$cbr8 set random_ null
$ns at 1.0 "$cbr8 start"
$ns at 99.0 "$cbr8 stop"

#Setup a CBR Application over UDP connection
set cbr9 [new Application/Traffic/CBR]
$cbr9 attach-agent $udp13
$cbr9 set packetSize_ 1000
$cbr9 set rate_ 1.0Mb
$cbr9 set random_ null
$ns at 1.0 "$cbr9 start"
$ns at 99.0 "$cbr9 stop"

#Setup a CBR Application over UDP connection
set cbr10 [new Application/Traffic/CBR]
$cbr10 attach-agent $udp14
$cbr10 set packetSize_ 1000
$cbr10 set rate_ 1.0Mb
$cbr10 set random_ null
$ns at 1.0 "$cbr10 start"
```

```
$ns at 99.0 "$cbr10 stop"

#Setup a CBR Application over UDP connection
set cbr11 [new Application/Traffic/CBR]
$cbr11 attach-agent $udp15
$cbr11 set packetSize_ 1000
$cbr11 set rate_ 1.0Mb
$cbr11 set random_ null
$ns at 1.0 "$cbr11 start"
$ns at 99.0 "$cbr11 stop"

#Setup a CBR Application over UDP connection
set cbr12 [new Application/Traffic/CBR]
$cbr12 attach-agent $udp16
$cbr12 set packetSize_ 1000
$cbr12 set rate_ 1.0Mb
$cbr12 set random_ null
$ns at 1.0 "$cbr12 start"
$ns at 99.0 "$cbr12 stop"


#===================================
#        Termination
#===================================
#Define a 'finish' procedure
proc finish {} {
    global ns tracefile namfile
    $ns flush-trace
    close $tracefile
    close $namfile
    exec nam out.nam &
    exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
    $ns at $val(stop) "\$n$i reset"
}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 5: Scenario 5 / TCL script 3 (Mesh Topology 50 nodes)

C. . Mesh Topology 50 nodes

```
#===================================
#     Simulation parameters setup
#===================================
set val(chan)   Channel/WirelessChannel    ;# channel type
set val(prop)   Propagation/TwoRayGround   ;# radio-propagation model
set val(netif)  Phy/WirelessPhy            ;# network interface type
set val(mac)    Mac/802_11                 ;# MAC type
set val(ifq)    Queue/DropTail/PriQueue    ;# interface queue type
set val(ll)     LL                         ;# link layer type
set val(ant)    Antenna/OmniAntenna        ;# antenna model
set val(ifqlen) 50                         ;# max packet in ifq
set val(nn)     50                         ;# number of mobilenodes
set val(rp)     DSDV                       ;# routing protocol
set val(x)      1839                       ;# X dimension of topography
set val(y)      100                        ;# Y dimension of topography
set val(stop)   100.0                      ;# time of simulation end

#===================================
#        Initialization
#===================================
#Create a ns simulator
set ns [new Simulator]

#Setup topography object
set topo       [new Topography]
$topo load_flatgrid $val(x) $val(y)
create-god $val(nn)

#Open the NS trace file
set tracefile [open out.tr w]
$ns trace-all $tracefile

#Open the NAM trace file
set namfile [open out.nam w]
$ns namtrace-all $namfile
$ns namtrace-all-wireless $namfile $val(x) $val(y)
set chan [new $val(chan)];#Create wireless channel

#===================================
#     Mobile node parameter setup
#===================================
$ns node-config -adhocRouting  $val(rp) \
```

```
                -llType        $val(ll) \
                -macType       $val(mac) \
                -ifqType       $val(ifq) \
                -ifqLen        $val(ifqlen) \
                -antType       $val(ant) \
                -propType      $val(prop) \
                -phyType       $val(netif) \
                -channel       $chan \
                -topoInstance  $topo \
                -agentTrace    ON \
                -routerTrace   ON \
                -macTrace      ON \
                -movementTrace ON

#================================
#       Nodes Definition
#================================
#Create 50 nodes
set n0 [$ns node]
$n0 set X_ 300
$n0 set Y_ 499
$n0 set Z_ 0.0
$ns initial_node_pos $n0 20
set n1 [$ns node]
$n1 set X_ 401
$n1 set Y_ 499
$n1 set Z_ 0.0
$ns initial_node_pos $n1 20
set n2 [$ns node]
$n2 set X_ 500
$n2 set Y_ 500
$n2 set Z_ 0.0
$ns initial_node_pos $n2 20
set n3 [$ns node]
$n3 set X_ 600
$n3 set Y_ 500
$n3 set Z_ 0.0
$ns initial_node_pos $n3 20
set n4 [$ns node]
$n4 set X_ 703
$n4 set Y_ 500
$n4 set Z_ 0.0
$ns initial_node_pos $n4 20
set n5 [$ns node]
$n5 set X_ 799
$n5 set Y_ 499
```

```
$n5 set Z_ 0.0
$ns initial_node_pos $n5 20
set n6 [$ns node]
$n6 set X_ 900
$n6 set Y_ 499
$n6 set Z_ 0.0
$ns initial_node_pos $n6 20
set n7 [$ns node]
$n7 set X_ 200
$n7 set Y_ 400
$n7 set Z_ 0.0
$ns initial_node_pos $n7 20
set n8 [$ns node]
$n8 set X_ 199
$n8 set Y_ 300
$n8 set Z_ 0.0
$ns initial_node_pos $n8 20
set n9 [$ns node]
$n9 set X_ 200
$n9 set Y_ 200
$n9 set Z_ 0.0
$ns initial_node_pos $n9 20
set n10 [$ns node]
$n10 set X_ 302
$n10 set Y_ 100
$n10 set Z_ 0.0
$ns initial_node_pos $n10 20
set n11 [$ns node]
$n11 set X_ 400
$n11 set Y_ 100
$n11 set Z_ 0.0
$ns initial_node_pos $n11 20
set n12 [$ns node]
$n12 set X_ 500
$n12 set Y_ 100
$n12 set Z_ 0.0
$ns initial_node_pos $n12 20
set n13 [$ns node]
$n13 set X_ 601
$n13 set Y_ 101
$n13 set Z_ 0.0
$ns initial_node_pos $n13 20
set n14 [$ns node]
$n14 set X_ 700
$n14 set Y_ 100
$n14 set Z_ 0.0
```

```
$ns initial_node_pos $n14 20
set n15 [$ns node]
$n15 set X_ 800
$n15 set Y_ 99
$n15 set Z_ 0.0
$ns initial_node_pos $n15 20
set n16 [$ns node]
$n16 set X_ 901
$n16 set Y_ 103
$n16 set Z_ 0.0
$ns initial_node_pos $n16 20
set n17 [$ns node]
$n17 set X_ 1000
$n17 set Y_ 201
$n17 set Z_ 0.0
$ns initial_node_pos $n17 20
set n18 [$ns node]
$n18 set X_ 1001
$n18 set Y_ 299
$n18 set Z_ 0.0
$ns initial_node_pos $n18 20
set n19 [$ns node]
$n19 set X_ 1002
$n19 set Y_ 399
$n19 set Z_ 0.0
$ns initial_node_pos $n19 20
set n20 [$ns node]
$n20 set X_ 1101
$n20 set Y_ 298
$n20 set Z_ 0.0
$ns initial_node_pos $n20 20
set n21 [$ns node]
$n21 set X_ 1102
$n21 set Y_ 200
$n21 set Z_ 0.0
$ns initial_node_pos $n21 20
set n22 [$ns node]
$n22 set X_ 101
$n22 set Y_ 299
$n22 set Z_ 0.0
$ns initial_node_pos $n22 20
set n23 [$ns node]
$n23 set X_ 98
$n23 set Y_ 200
$n23 set Z_ 0.0
$ns initial_node_pos $n23 20
```

```
set n24 [$ns node]
$n24 set X_ 1200
$n24 set Y_ 251
$n24 set Z_ 0.0
$ns initial_node_pos $n24 20
set n25 [$ns node]
$n25 set X_ 301
$n25 set Y_ 600
$n25 set Z_ 0.0
$ns initial_node_pos $n25 20
set n26 [$ns node]
$n26 set X_ 401
$n26 set Y_ 602
$n26 set Z_ 0.0
$ns initial_node_pos $n26 20
set n27 [$ns node]
$n27 set X_ 498
$n27 set Y_ 602
$n27 set Z_ 0.0
$ns initial_node_pos $n27 20
set n28 [$ns node]
$n28 set X_ 599
$n28 set Y_ 603
$n28 set Z_ 0.0
$ns initial_node_pos $n28 20
set n29 [$ns node]
$n29 set X_ 696
$n29 set Y_ 602
$n29 set Z_ 0.0
$ns initial_node_pos $n29 20
set n30 [$ns node]
$n30 set X_ 799
$n30 set Y_ 605
$n30 set Z_ 0.0
$ns initial_node_pos $n30 20
set n31 [$ns node]
$n31 set X_ 898
$n31 set Y_ 600
$n31 set Z_ 0.0
$ns initial_node_pos $n31 20
set n32 [$ns node]
$n32 set X_ 1002
$n32 set Y_ 503
$n32 set Z_ 0.0
$ns initial_node_pos $n32 20
set n33 [$ns node]
```

```
$n33 set X_ 1102
$n33 set Y_ 400
$n33 set Z_ 0.0
$ns initial_node_pos $n33 20
set n34 [$ns node]
$n34 set X_ 1099
$n34 set Y_ 103
$n34 set Z_ 0.0
$ns initial_node_pos $n34 20
set n35 [$ns node]
$n35 set X_ 998
$n35 set Y_ 106
$n35 set Z_ 0.0
$ns initial_node_pos $n35 20
set n36 [$ns node]
$n36 set X_ 998
$n36 set Y_ 5
$n36 set Z_ 0.0
$ns initial_node_pos $n36 20
set n37 [$ns node]
$n37 set X_ 901
$n37 set Y_ 2
$n37 set Z_ 0.0
$ns initial_node_pos $n37 20
set n38 [$ns node]
$n38 set X_ 798
$n38 set Y_ 5
$n38 set Z_ 0.0
$ns initial_node_pos $n38 20
set n39 [$ns node]
$n39 set X_ 704
$n39 set Y_ 0
$n39 set Z_ 0.0
$ns initial_node_pos $n39 20
set n40 [$ns node]
$n40 set X_ 599
$n40 set Y_ 3
$n40 set Z_ 0.0
$ns initial_node_pos $n40 20
set n41 [$ns node]
$n41 set X_ 499
$n41 set Y_ -1
$n41 set Z_ 0.0
$ns initial_node_pos $n41 20
set n42 [$ns node]
$n42 set X_ 399
```

```
$n42 set Y_ -1
$n42 set Z_ 0.0
$ns initial_node_pos $n42 20
set n43 [$ns node]
$n43 set X_ 299
$n43 set Y_ 3
$n43 set Z_ 0.0
$ns initial_node_pos $n43 20
set n44 [$ns node]
$n44 set X_ 201
$n44 set Y_ 6
$n44 set Z_ 0.0
$ns initial_node_pos $n44 20
set n45 [$ns node]
$n45 set X_ 199
$n45 set Y_ 103
$n45 set Z_ 0.0
$ns initial_node_pos $n45 20
set n46 [$ns node]
$n46 set X_ 101
$n46 set Y_ 99
$n46 set Z_ 0.0
$ns initial_node_pos $n46 20
set n47 [$ns node]
$n47 set X_ 1
$n47 set Y_ 202
$n47 set Z_ 0.0
$ns initial_node_pos $n47 20
set n48 [$ns node]
$n48 set X_ 0
$n48 set Y_ 303
$n48 set Z_ 0.0
$ns initial_node_pos $n48 20
set n49 [$ns node]
$n49 set X_ 99
$n49 set Y_ 400
$n49 set Z_ 0.0
$ns initial_node_pos $n49 20

#===================================
#        Agents Definition
#===================================
#Setup a UDP connection
set udp2 [new Agent/UDP]
$ns attach-agent $n22 $udp2
set null0 [new Agent/Null]
```

```
$ns attach-agent $n8 $null0
$ns connect $udp2 $null0
$udp2 set packetSize_ 1500

#Setup a UDP connection
set udp3 [new Agent/UDP]
$ns attach-agent $n23 $udp3
set null29 [new Agent/Null]
$ns attach-agent $n8 $null29
$ns connect $udp3 $null29
$udp3 set packetSize_ 1500

#Setup a UDP connection
set udp4 [new Agent/UDP]
$ns attach-agent $n9 $udp4
set null28 [new Agent/Null]
$ns attach-agent $n8 $null28
$ns connect $udp4 $null28
$udp4 set packetSize_ 1500

#Setup a UDP connection
set udp5 [new Agent/UDP]
$ns attach-agent $n7 $udp5
set null17 [new Agent/Null]
$ns attach-agent $n8 $null17
$ns connect $udp5 $null17
$udp5 set packetSize_ 1500

#Setup a UDP connection
set udp6 [new Agent/UDP]
$ns attach-agent $n0 $udp6
set null18 [new Agent/Null]
$ns attach-agent $n8 $null18
$ns connect $udp6 $null18
$udp6 set packetSize_ 1500

#Setup a UDP connection
set udp7 [new Agent/UDP]
$ns attach-agent $n1 $udp7
set null19 [new Agent/Null]
$ns attach-agent $n8 $null19
$ns connect $udp7 $null19
$udp7 set packetSize_ 1500

#Setup a UDP connection
set udp8 [new Agent/UDP]
```

```
$ns attach-agent $n2 $udp8
set null20 [new Agent/Null]
$ns attach-agent $n8 $null20
$ns connect $udp8 $null20
$udp8 set packetSize_ 1500

#Setup a UDP connection
set udp9 [new Agent/UDP]
$ns attach-agent $n3 $udp9
set null21 [new Agent/Null]
$ns attach-agent $n5 $null21
$ns connect $udp9 $null21
$udp9 set packetSize_ 1500

#Setup a UDP connection
set udp10 [new Agent/UDP]
$ns attach-agent $n4 $udp10
set null22 [new Agent/Null]
$ns attach-agent $n5 $null22
$ns connect $udp10 $null22
$udp10 set packetSize_ 1500

#Setup a UDP connection
set udp11 [new Agent/UDP]
$ns attach-agent $n6 $udp11
set null24 [new Agent/Null]
$ns attach-agent $n5 $null24
$ns connect $udp11 $null24
$udp11 set packetSize_ 1500

#Setup a UDP connection
set udp12 [new Agent/UDP]
$ns attach-agent $n19 $udp12
set null23 [new Agent/Null]
$ns attach-agent $n5 $null23
$ns connect $udp12 $null23
$udp12 set packetSize_ 1500

#Setup a UDP connection
set udp13 [new Agent/UDP]
$ns attach-agent $n18 $udp13
set null25 [new Agent/Null]
$ns attach-agent $n5 $null25
$ns connect $udp13 $null25
$udp13 set packetSize_ 1500
```

```
#Setup a UDP connection
set udp14 [new Agent/UDP]
$ns attach-agent $n15 $udp14
set null26 [new Agent/Null]
$ns attach-agent $n5 $null26
$ns connect $udp14 $null26
$udp14 set packetSize_ 1500

#Setup a UDP connection
set udp15 [new Agent/UDP]
$ns attach-agent $n13 $udp15
set null27 [new Agent/Null]
$ns attach-agent $n5 $null27
$ns connect $udp15 $null27
$udp15 set packetSize_ 1500

#Setup a UDP connection
set udp16 [new Agent/UDP]
$ns attach-agent $n11 $udp16
set null1 [new Agent/Null]
$ns attach-agent $n5 $null1
$ns connect $udp16 $null1
$udp16 set packetSize_ 1500

#Setup a UDP connection
set udp36 [new Agent/UDP]
$ns attach-agent $n41 $udp36
set null34 [new Agent/Null]
$ns attach-agent $n36 $null34
$ns connect $udp36 $null34
$udp36 set packetSize_ 1500

#Setup a UDP connection
set udp37 [new Agent/UDP]
$ns attach-agent $n43 $udp37
set null35 [new Agent/Null]
$ns attach-agent $n36 $null35
$ns connect $udp37 $null35
$udp37 set packetSize_ 1500

#Setup a UDP connection
set udp38 [new Agent/UDP]
$ns attach-agent $n47 $udp38
set null31 [new Agent/Null]
$ns attach-agent $n36 $null31
$ns connect $udp38 $null31
```

```
$udp38 set packetSize_ 1500

#Setup a UDP connection
set udp39 [new Agent/UDP]
$ns attach-agent $n27 $udp39
set null30 [new Agent/Null]
$ns attach-agent $n25 $null30
$ns connect $udp39 $null30
$udp39 set packetSize_ 1500

#Setup a UDP connection
set udp40 [new Agent/UDP]
$ns attach-agent $n29 $udp40
set null32 [new Agent/Null]
$ns attach-agent $n25 $null32
$ns connect $udp40 $null32
$udp40 set packetSize_ 1500

#Setup a UDP connection
set udp41 [new Agent/UDP]
$ns attach-agent $n31 $udp41
set null33 [new Agent/Null]
$ns attach-agent $n25 $null33
$ns connect $udp41 $null33
$udp41 set packetSize_ 1500


#==================================
#        Applications Definition
#==================================
#Setup a CBR Application over UDP connection
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp2
$cbr0 set packetSize_ 1000
$cbr0 set rate_ 1.0Mb
$cbr0 set random_ null
$ns at 1.0 "$cbr0 start"
$ns at 99.0 "$cbr0 stop"

#Setup a CBR Application over UDP connection
set cbr1 [new Application/Traffic/CBR]
$cbr1 attach-agent $udp5
$cbr1 set packetSize_ 1000
$cbr1 set rate_ 1.0Mb
$cbr1 set random_ null
$ns at 1.0 "$cbr1 start"
```

```
$ns at 99.0 "$cbr1 stop"

#Setup a CBR Application over UDP connection
set cbr2 [new Application/Traffic/CBR]
$cbr2 attach-agent $udp6
$cbr2 set packetSize_ 1000
$cbr2 set rate_ 1.0Mb
$cbr2 set random_ null
$ns at 1.0 "$cbr2 start"
$ns at 99.0 "$cbr2 stop"

#Setup a CBR Application over UDP connection
set cbr3 [new Application/Traffic/CBR]
$cbr3 attach-agent $udp7
$cbr3 set packetSize_ 1000
$cbr3 set rate_ 1.0Mb
$cbr3 set random_ null
$ns at 1.0 "$cbr3 start"
$ns at 99.0 "$cbr3 stop"

#Setup a CBR Application over UDP connection
set cbr4 [new Application/Traffic/CBR]
$cbr4 attach-agent $udp8
$cbr4 set packetSize_ 1000
$cbr4 set rate_ 1.0Mb
$cbr4 set random_ null
$ns at 1.0 "$cbr4 start"
$ns at 99.0 "$cbr4 stop"

#Setup a CBR Application over UDP connection
set cbr5 [new Application/Traffic/CBR]
$cbr5 attach-agent $udp9
$cbr5 set packetSize_ 1000
$cbr5 set rate_ 1.0Mb
$cbr5 set random_ null
$ns at 1.0 "$cbr5 start"
$ns at 99.0 "$cbr5 stop"

#Setup a CBR Application over UDP connection
set cbr6 [new Application/Traffic/CBR]
$cbr6 attach-agent $udp10
$cbr6 set packetSize_ 1000
$cbr6 set rate_ 1.0Mb
$cbr6 set random_ null
$ns at 1.0 "$cbr6 start"
$ns at 99.0 "$cbr6 stop"
```

```
#Setup a CBR Application over UDP connection
set cbr7 [new Application/Traffic/CBR]
$cbr7 attach-agent $udp11
$cbr7 set packetSize_ 1000
$cbr7 set rate_ 1.0Mb
$cbr7 set random_ null
$ns at 1.0 "$cbr7 start"
$ns at 99.0 "$cbr7 stop"

#Setup a CBR Application over UDP connection
set cbr8 [new Application/Traffic/CBR]
$cbr8 attach-agent $udp12
$cbr8 set packetSize_ 1000
$cbr8 set rate_ 1.0Mb
$cbr8 set random_ null
$ns at 1.0 "$cbr8 start"
$ns at 99.0 "$cbr8 stop"

#Setup a CBR Application over UDP connection
set cbr9 [new Application/Traffic/CBR]
$cbr9 attach-agent $udp13
$cbr9 set packetSize_ 1000
$cbr9 set rate_ 1.0Mb
$cbr9 set random_ null
$ns at 1.0 "$cbr9 start"
$ns at 99.0 "$cbr9 stop"

#Setup a CBR Application over UDP connection
set cbr10 [new Application/Traffic/CBR]
$cbr10 attach-agent $udp14
$cbr10 set packetSize_ 1000
$cbr10 set rate_ 1.0Mb
$cbr10 set random_ null
$ns at 1.0 "$cbr10 start"
$ns at 99.0 "$cbr10 stop"

#Setup a CBR Application over UDP connection
set cbr11 [new Application/Traffic/CBR]
$cbr11 attach-agent $udp15
$cbr11 set packetSize_ 1000
$cbr11 set rate_ 1.0Mb
$cbr11 set random_ null
$ns at 1.0 "$cbr11 start"
$ns at 99.0 "$cbr11 stop"
```

```
#Setup a CBR Application over UDP connection
set cbr12 [new Application/Traffic/CBR]
$cbr12 attach-agent $udp16
$cbr12 set packetSize_ 1000
$cbr12 set rate_ 1.0Mb
$cbr12 set random_ null
$ns at 1.0 "$cbr12 start"
$ns at 99.0 "$cbr12 stop"

#Setup a CBR Application over UDP connection
set cbr13 [new Application/Traffic/CBR]
$cbr13 attach-agent $udp39
$cbr13 set packetSize_ 1000
$cbr13 set rate_ 1.0Mb
$cbr13 set random_ null
$ns at 1.0 "$cbr13 start"
$ns at 99.0 "$cbr13 stop"

#Setup a CBR Application over UDP connection
set cbr14 [new Application/Traffic/CBR]
$cbr14 attach-agent $udp40
$cbr14 set packetSize_ 1000
$cbr14 set rate_ 1.0Mb
$cbr14 set random_ null
$ns at 1.0 "$cbr14 start"
$ns at 99.0 "$cbr14 stop"

#Setup a CBR Application over UDP connection
set cbr15 [new Application/Traffic/CBR]
$cbr15 attach-agent $udp41
$cbr15 set packetSize_ 1000
$cbr15 set rate_ 1.0Mb
$cbr15 set random_ null
$ns at 1.0 "$cbr15 start"
$ns at 99.0 "$cbr15 stop"

#Setup a CBR Application over UDP connection
set cbr16 [new Application/Traffic/CBR]
$cbr16 attach-agent $udp36
$cbr16 set packetSize_ 1000
$cbr16 set rate_ 1.0Mb
$cbr16 set random_ null
$ns at 1.0 "$cbr16 start"
$ns at 99.0 "$cbr16 stop"

#Setup a CBR Application over UDP connection
```

```
set cbr17 [new Application/Traffic/CBR]
$cbr17 attach-agent $udp37
$cbr17 set packetSize_ 1000
$cbr17 set rate_ 1.0Mb
$cbr17 set random_ null
$ns at 1.0 "$cbr17 start"
$ns at 99.0 "$cbr17 stop"

#Setup a CBR Application over UDP connection
set cbr18 [new Application/Traffic/CBR]
$cbr18 attach-agent $udp38
$cbr18 set packetSize_ 1000
$cbr18 set rate_ 1.0Mb
$cbr18 set random_ null
$ns at 1.0 "$cbr18 start"
$ns at 99.0 "$cbr18 stop"


#=================================
#       Termination
#=================================
#Define a 'finish' procedure
proc finish {} {
   global ns tracefile namfile
   $ns flush-trace
   close $tracefile
   close $namfile
   exec nam out.nam &
   exit 0
}
for {set i 0} {$i < $val(nn) } { incr i } {
   $ns at $val(stop) "\$n$i reset"

}
$ns at $val(stop) "$ns nam-end-wireless $val(stop)"
$ns at $val(stop) "finish"
$ns at $val(stop) "puts \"done\" ; $ns halt"
$ns run
```

Appendix 6: Résumé en français

Introduction

De nos jours, la technologie des réseaux de capteurs sans fil est devenue extrêmement vitale dans une diversité d'aspects. Elle est devenu nécessaire dans de nombreux domaines où l'utilisation de cette technologie de pointe s'est transformée non seulement en une option mais aussi en une obligation, Dans l'environnement de santé ou dans l'utilisation militaire.

En effet, il existe de nombreuses technologies de capteurs sans fil qui diffèrent en termes de domaine d'utilisation, tel que les systèmes de capteurs sans fil Zigbee qui sont plus efficaces pour la consommation d'énergie, ou les réseaux de capteurs sans fil Wi-Fi qui offrent des débits de données plus élevés et des performances améliorées. Ils ont leur domaine d'application particulier.

Mis à part cela, ce mémoire de maîtrise contiendra quatre autres chapitres. Au début, le chapitre sur l'état de l'art qui va souligner les points théoriques importants qui seront nécessaires pour accomplir ce travail et qui ont été utilisés dans ce projet de recherche, ce chapitre donnera un point de Vue théorique du besoin du Réseau de capteurs sans fil ainsi que la théorie derrière diverses technologies WSN, en particulier les protocoles de routage.

Ensuite, dans le chapitre suivant, les principaux résultats relatifs de la recherche seront discutés; Ensuite, à partir de cette discussion, nous pourrons conclure si le nombre de nœuds utilisés dans le WSN affecte ou non le fonctionnement du réseau, et quel protocole de routage fonctionne mieux que l'autre et ainsi de suite.

Ce chapitre inclura également quelques figures Et des tableaux qui seront présentés pour montrer les étapes de test et de simulation ainsi que les résultats.

Et enfin, dans le chapitre conclusion, un résumé de ce qui a été effectué tout au long de ce rapport sera présenté; Aussi, certaines recommandations qui pourraient améliorer la performance du WSN seront fournies, de sorte que cela aidera vraisemblablement à poursuivre les recherches dans le domaine de WSN.

1. Objectifs du projet

L'objectif de ce projet de recherche est de concevoir et simuler différentes topologies WSN pour la transmission de données vidéo dans le contexte des lampadaires intelligents. Des méthodes de routage seront comparé pour atteindre le débit requis et pour respecter les contraintes de déploiement spécifiques. Des simulations de réseau seront réalisées pour valider l'étude théorique.

2. Approche de recherche

Pour ce projet de maîtrise, l'approche de la recherche est fondamentalement une recherche basée sur des travaux antérieurs, tout en incluant certains aspects de recherche basés sur la simulation et la comparaison des différents résultats, car nous allons concevoir, construire et tester différentes topologies de réseau de capteurs sans fil.

En outre, la recherche scientifique est définie comme suit: étude d'un sujet en détail pour découvrir de nouvelles informations ou pour découvrir une nouvelle interprétation [12]. En outre, pour ce projet final, des approches quantitatives et qualitatives seront utilisées et, fondamentalement, le projet sera basé sur un aspect de recherche primaire et originale.

3. Méthodologie

NS2 ou Network Simulator 2, est essentiellement un logiciel de simulation basé sur les événements, il est largement utilisé dans l'expérimentation de réseaux de communication.

NS2 offre la possibilité de simuler des réseaux câblés et sans fil ainsi que leurs protocoles spécifiques [14].

NS2 a été créé en 1989, et comme il est très flexible et composé de plusieurs unités qui peuvent y être ajoutées, NS2 est devenu largement apprécié dans la communauté de chercheurs en réseau [14].

L'Université Cornell et l'Université de Californie ont contribué de manière significative à l'évolution du Network Simulator et, en 1995, DARPA a contribué aussi à son amélioration [14].

Récemment, la Fondation de sciences National «National Science Foundation» est devenue un contributeur à la progression NS2.

3.1 Architecture de base de NS2

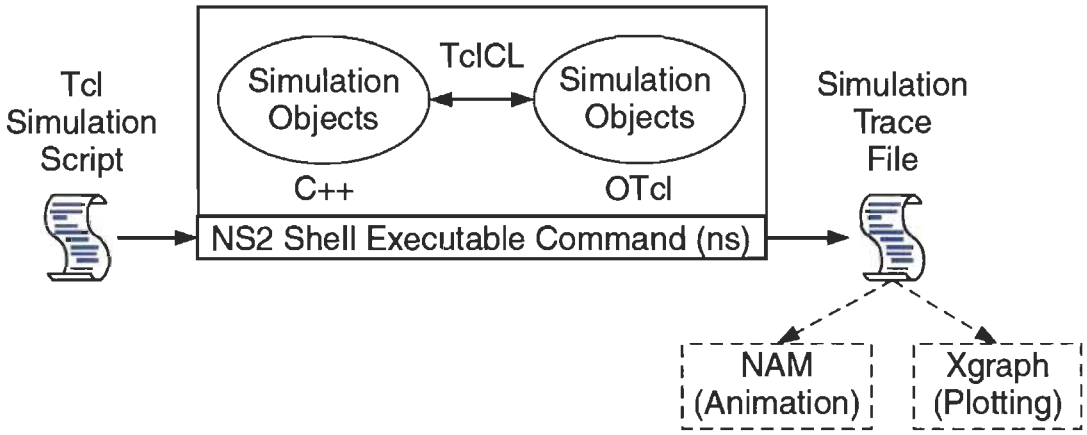


Figure 35 Architecture fondamentale du Network Simulator NS [14]

Cette figure ci-dessus montre l'architecture fondamentale de NS2. NS2 offre une commande nommée "ns" pour exécuter le script TCL.

Par exemple, pour exécuter un script TCL nommé "Exemple1.Tcl", nous devons rédiger la commande suivante "ns Exemple1.Tcl"

Ensuite, si la commande est exécutée correctement, un fichier de trace de simulation sera généré, et il est essentiel pour tracer un graphique et d'analyser le comportement du réseau.

NS2 est fondé autour de deux langages de programmation principaux le langage C ++ et le langage OTcl (commande d'outil orientée objet). C ++ décrit le mécanisme intérieur des objets de simulation, l'OTcl désigne le mécanisme extérieur tel que la planification d'évènements discrets, la construction et la conception des objets [14].

TclCL se connecte entre les deux langages de programmation (OTcl et C ++). Les variables définies dans un OTcl sont liées à des objets C ++. Cette variable est en fait une chaîne dans l'OTcl et ne détient aucun rôle, mais le rôle est décrit dans l'objet C ++ lié. Dans OTcl, les variables se comportent comme une interface qui communique avec les utilisateurs et d'autres objets OTcl [14].

3.2 Installation

NS2 est un outil de simulation open source, et il peut être téléchargé gratuitement sur son site officiel ns.com. Même si il a été établie dans l'Écosystème UNIX, NS2 est capable de fonctionner sur plusieurs systèmes d'exploitation tels que Unix, Mac et Windows.

Dans notre cas, nous avons choisi d'installer NS2 version 2.35 sur Linux Ubuntu

12.04 LTS car cela fonctionne très bien sur ce système d'exploitation.

Table 3.3 Liste des étapes pour installer NS2

Steps	Description
1- Sudo apt-get update	Pour installer les mises à jour nécessaires pour le système d'exploitation
2- Sudo apt-get install build essential autoconf automake libxmu-dev	Pour installer la bibliothèque appelée "libxmu-dev" sur le système d'exploitation,
3- Download NS2.35	Téléchargez le fichier zippé NS2
4- tar zxvf ns-allinone-2.35.tar.gz	Décompressez le fichier zippé NS2
5- cd ns-allinone-2.35	Accéder au répertoire de NS2
6- ./install	Installer NS2
7- ./validate	Vérifiez si chaque composant est correctement installé

3.3 Dossiers et convention

Maintenant, NS2 est installé dans le répertoire nsallinone-2.35. La figure ci-dessous illustre la structure du répertoire sous nsallinone-2.35.

Comme le montre la figure ci-dessous, il existe quatre niveaux de répertoire. Tout d'abord, le répertoire nsallinone-2.35, puis au niveau deux, nous trouvons les modules de simulation NS2 et les classes TclCL. Ensuite, dans le niveau

trois, il existe des modules dans la hiérarchie interprétée et finalement au quatrième niveau, ce sont les modules fréquemment utilisés.

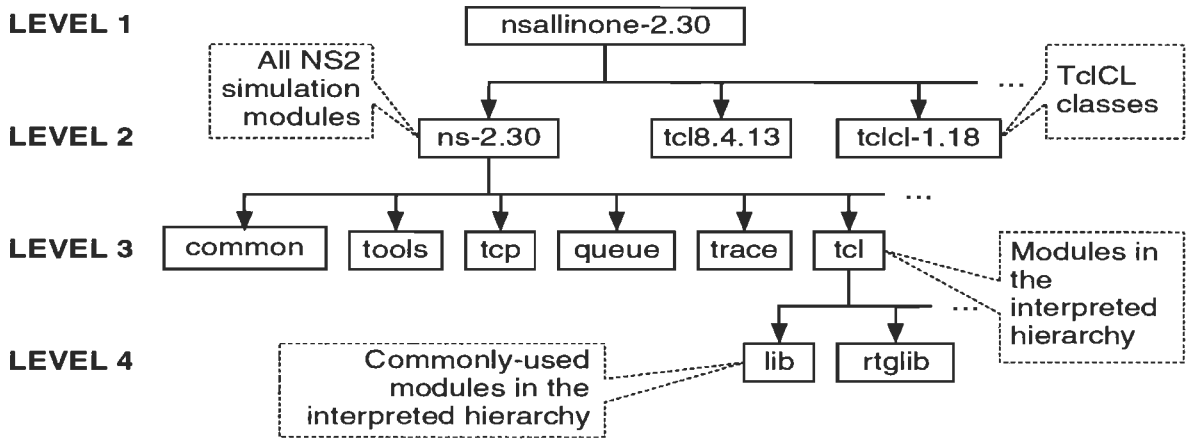


Figure 36 Structure de répertoire sous NS2 tout en un Package [14].

3.4 Running NS2 Simulation

3.4.1 NS2 Program invocation

Comme NS2 est maintenant installé, il peut être invoqué à l'aide de cette commande "NS [nomfichier.tcl] [arg]"

[Nom_fichier.tcl] [arg] ne sont pas des arguments obligatoires. Dans ce cas, si la commande "NS" n'a pas reçu d'argument, un domaine NS2 sera invoqué et NS2 sera prêt à exécuter les commandes dès qu'ils seront écrits. Et si, en plus de la recommandation "NS", l'argenment "[filename.tcl]" sera donné, NS2 exécutera l'ensemble du script TCL.

3.4.2 Main NS2 simulation steps

Table 3.4 Principales étapes de simulation NS2

Simulation Steps	Description
1- Conception	Conception du scénario de simulation
2- Écrire le programme	Écrire le script TCL
3- Exécution, tester et reconfiguration	Exécutez le script TCL
4- Traitement de post-simulation	Analyser le fichier de trace (out.tr) afin de comprendre le comportement du réseau.

Le tableau ci-dessus résume les quatre étapes principales de la simulation NS2, Premièrement, dans l'étape de conception, le but et le report du réseau doivent être réparés tant que les performances du réseau et la configuration, d'autre part, le script TCL doit être écrit conformément à la Conception du réseau, puis dans la troisième étape, le script TCL devrait être exécuté pour le tester et le reconfigurer si nécessaire. Et enfin, le fichier de trace (out.tr) doit être analysé pour décrire le comportement du réseau et pour observer ses performances.

Scenario 1 : Comparison of Mesh topology 50 nodes with different routing protocols (AODV, DSR, DSDV) (Wi-Fi)

Throughput Mesh topology 50 nodes with different routing protocols (AODV, DSR, DSDV) (Wi-Fi)

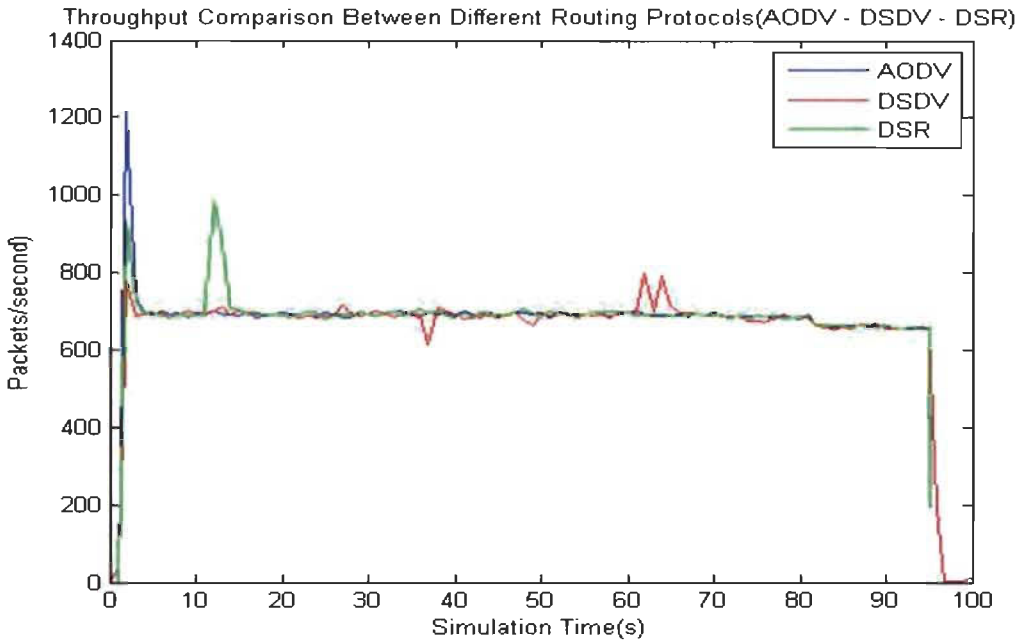


Figure 37 Comparaison de débit entre différents protocoles de routage (AODV, DSDV, DSR)

Dans ce scénario, nous allons comparer ces paramètres de performance: débit, gigue (Jitter) et délai de bout en bout (End-End delay) du Mesh topology (50 noeuds) avec différents protocoles de routage (AODV, DSR, DSDV) afin de découvrir quel protocole de routage fonctionne le mieux.

Dans la figure ci-dessus, nous avons comparé le débit de différents protocoles de routage des noeuds AODV, DSR et DSDV.

Comme il est indiqué dans le tableau ci-dessus, la configuration du scénario de simulation fonctionne à 1 Mbps (débit de données) pour les trois senarios DSDV AODV et DSR.

Comme il est indiqué dans la figure ci-dessus, nous constatons que le débit le plus élevé lorsque nous avons utilisé le protocole de routage AODV. Dans les premières secondes de la simulation, il a chuté pour atteindre environ 700 paquets par seconde.

Ensuite, en arrivant à la 4ème seconde, tous les protocoles de routage étaient assez stables dans environ 700 paquets par seconde, puis durant 12ème seconde le débit de DSR a augmenté de façon brusque pour atteindre environ 1000 paquets par seconde.

Le débit de DSDV a montré moins de stabilité que les autres puisque, en atteignant la 35ème seconde, il a diminué pour atteindre 600 paquets par seconde puis entre 60 et 70 secondes, il a fluctué entre 800 et 700 paquets par seconde.

Lorsqu'il y a un gros trafic, le taux de collision augmente et c'est ainsi que le débit du système est affecté. Nous observons que le débit le plus élevé lors de l'utilisation de AODV était d'environ 1200 paquets par seconde à la troisième seconde et pendant la plupart du temps de la simulation, le débit est resté stable à environ 700 paquets par seconde jusqu'à la fin de la simulation.

Scénario 2 : Topologie Maillé (Mesh) 50 nodes (Wi-Fi) Comparaison entre différents formats vidéo (CIF, 4CIF, QCIF ...) avec protocole de routage DSDV

Débit du topologie Maillé (Mesh) avec 50 nodes (Wi-Fi) Comparaison entre différents formats vidéo avec protocole de routage DSDV

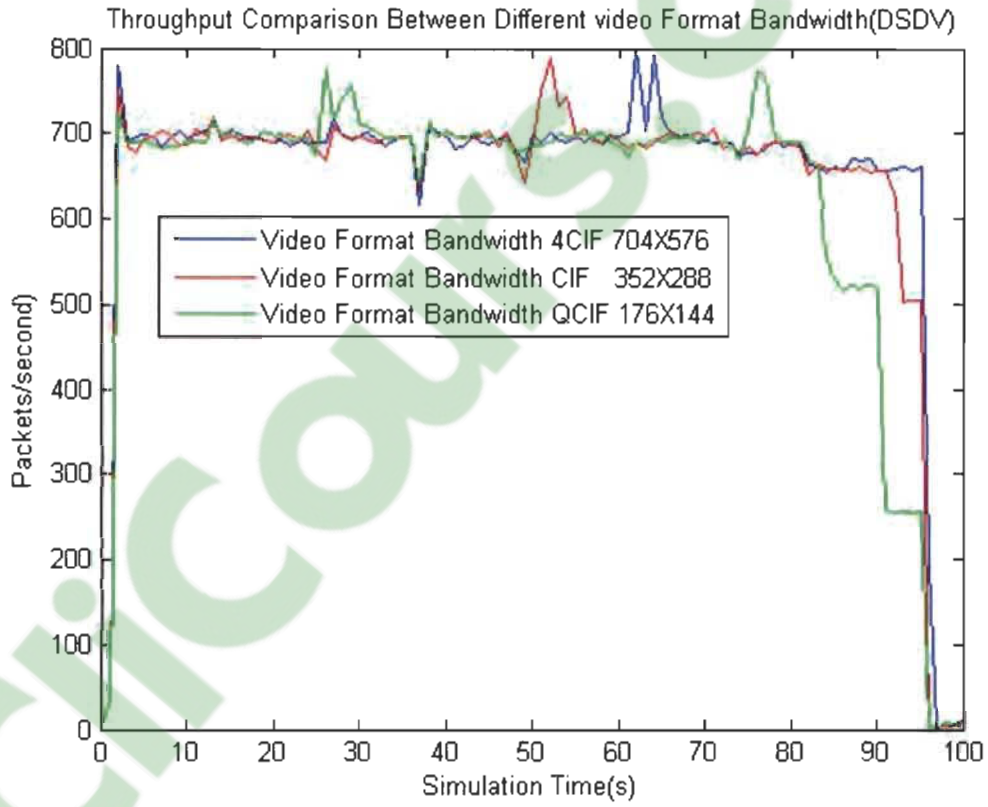


Figure 38 Débit: topologie de maillage 50 noeuds (Wi-Fi) Comparaison entre différents formats vidéo avec protocole de routage DSDV

Afin d'avoir une idée sur le fonctionnement du réseau, l'analyse du débit du réseau est parmi les paramètres les plus populaire. Pas comme dans le cas de

la transmission de messages texte, une application vidéo peut nécessiter beaucoup plus de bande passante.

Dans la figure ci-dessus, nous avons comparé le débit de différents formats vidéo 4CIF (704X576) CIF (352X288) et QCIF (176X144).
La configuration du scénario de simulation fonctionnant à ces débits de données suivants 260 Kb / s, 512 kb / s, 1 Mb / s pour ces trois scénarios avec le format vidéo 4CIF (704X576), CIF (352X288) et QCIF (176X144).
En observant la la figure ci-dessus, nous avons remarqué que, avec le format vidéo 4CIF, le débit le plus élevé atteint était d'environ 800 paquets par seconde et le débit le plus bas atteint était de 600 paquets par seconde et, à environ 700 paquets par seconde pour la plupart du temps de simulation.

D'autre part, avec le format de vidéo CIF, le débit le plus élevé atteint était de moins de 800 paquets par seconde et le débit le plus bas atteint était de 650 paquets par seconde et il était stable à environ 700 paquets par seconde pour la plupart du temps de simulation.

Scenario 3 : Comparaison entre les deux technologies Wi-Fi et Zigbee avec Topologie Maillé (Mesh) 50 nœuds (avec AODV comme protocole de routage)

Débit du Résultat Wi-Fi vs Résultat Zigbee 50 nœuds Topologie Mesh (avec AODV)

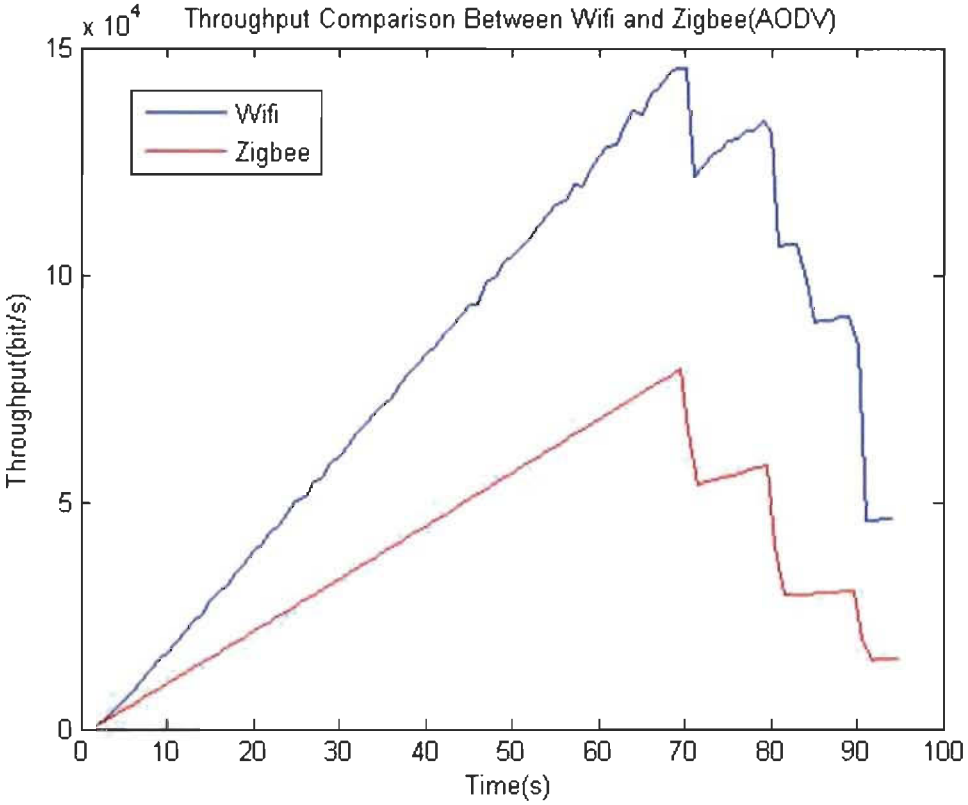


Figure 39 Débit: Comparaison entre les deux technologies Wi-Fi et Zigbee avec Topologie Maillé (Mesh) 50 nœuds (with AODV)

La configuration du scénario de simulation est résumée dans les points suivants. Les débits de données 260 Kb / s pour les deux senarios Wi-Fi et Zigbee avec le

même nombre de nœuds (50 noeuds) et la même Topologie de réseau (topologie maillé) et avec le même Protocole de routage (AODV).

Comme il est indiqué dans la figure ci-dessus, nous avons remarqué qu'avec le protocole Wi-Fi 802.11, le débit le plus élevé atteint était d'environ 150 Kb / s paquets par seconde et le débit le plus bas atteint était de 50 Kb / s, après environ 70 secondes, la courbe a commencé de diminuer de manière constante jusqu'à atteindre 50 Kb / s à la fin du temps de simulation.
D'autre part, avec le protocole Zigbee 802.15.4, le débit le plus élevé atteint était d'environ 70 Kb / s paquets par seconde et le débit le plus bas atteint était de 10 Kb / s, après environ 70 secondes, la courbe a commencé de diminuer jusqu'à atteindre 10 Kb / s à la fin du temps de simulation.

En outre, pour tous les formats vidéo, au début de la simulation, le débit a considérablement augmenté entre 0 et 70 secondes jusqu'à atteindre environ 150 Kb / s pour le protocole Wi-Fi 802.11 et 70 Kb / s pour le protocole Zigbee, puis après la septième seconde, il a fluctué Jusqu'à ce qu'il atteigne la fin de la simulation, ces fluctuations du débit est due aux hausses et aux chutes de collusion et de trafic à un moment précis.

Lorsqu'il y a un gros trafic, le taux de collision augmente et c'est ainsi que le débit du système est affecté. Ainsi, en termes de débit de données Le protocole Wi-Fi 802.11 a montré plus de capacité que le protocole Zigbee 802.15.4

Scénario 4 : Comparaison entre topologies (Topologie d'étoiles, topologie de maille et topologie de ligne) 11 noeuds (Wi-Fi) AODV

Comparaison de débit entre topologies (topologie étoile, topologie de maillage et topologie de ligne) 11 noeuds (Wi-Fi) AODV.

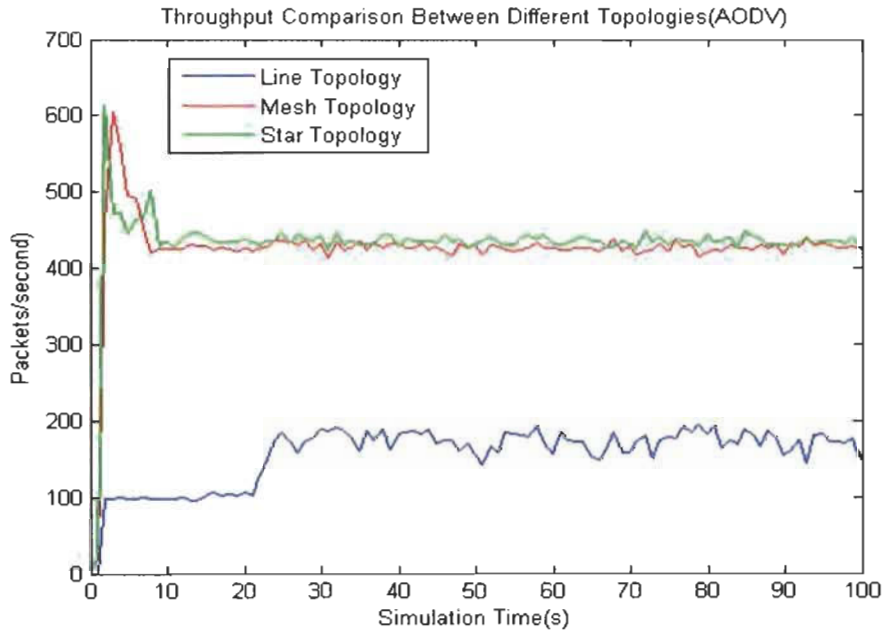


Figure 40 Comparaison de débit entre topologies (topologie étoile, topologie de maillage et topologie de ligne) 11 noeuds (Wi-Fi) AODV.

Dans la figure ci-dessus, nous avons comparé le débit de la topologie de réseau différente ligne (Line), Maillé (Mesh) et étoile (Star).

La configuration du scénario de simulation est résumée dans les points suivants. Le débit de données est fixé à 1500 b / s et le nombre de nœuds est de 11 nœuds, le protocole de transfert est UDP et CBR utilisé comme trafic de données et le protocole de routage Est AODV.

Comme le montre la figure ci-dessus, nous avons remarqué qu'avec la topologie en étoile, le débit le plus élevé atteint était d'environ 610 paquets par seconde et le débit le plus bas atteint était de 450 paquets par seconde et il s'est stabilisé à ce débit pour la plupart de temps de la simulation.

De plus, comme il est présenté dans la figure ci-dessus, nous avons remarqué qu'avec la topologie Mesh, le débit le plus élevé atteint était d'environ 600 paquets par seconde et le débit le plus bas atteint était d'environ 430 paquets par seconde et il s'est stabilisé à ce débit pour La plupart du temps de simulation.

En Plus, avec la topologie de ligne, le débit le plus élevé atteint était d'environ 200 paquets par seconde puis, il variait entre 150 et 200 paquets par seconde pour la plupart du temps de simulation.

En outre, pour tous les topologies, au début de la simulation, le débit a augmenté de façon brusque entre 0 et environ 3 secondes jusqu'à ce qu'il atteigne environ 100 paquets par seconde pour la topologie des lignes et légèrement plus de 600 paquets par seconde pour les topologies étoilées et mailles, Puis après la quatrième seconde, il a fluctué régulièrement jusqu'à ce qu'il atteigne la fin de la simulation. Et ces fluctuations du débit est due à l'augmentation et à la diminution de la collusion et du trafic à un moment précis.

Scénario 5 : Influence du nombre de nœuds sur la performance: comparaison du nombre de nœuds différents pour la topologie Mesh (11 noeuds, 20 noeuds, 50 nœuds) (Wi-Fi) (AODV)

Tableau 4.8 Scénario 5 Nombre de nœuds par délai moyen

Nombre de nœuds	Délai Moyen(ms)
11	4.25
25	1.86
50	2.61

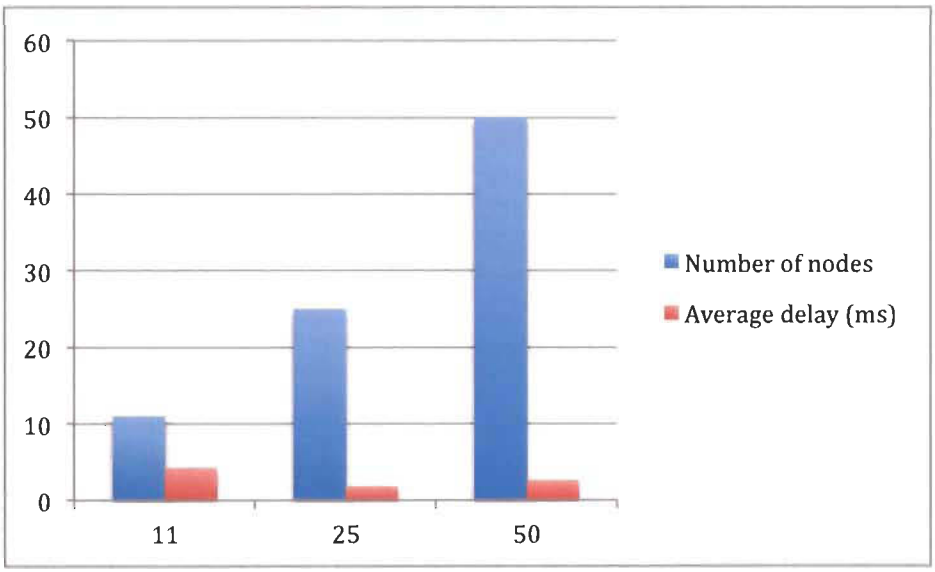


Figure 41 Bar Chart : Nombre de nœuds par débit moyen

Dans le scénario des 11 nœuds, 10 nœuds communiquent avec le nœud source et aussi dans le scénario des 25 nœuds 10 nœuds communiquent avec le nœud

source et dans le scénario de 50 nœuds, 25 nœuds communiquent avec le nœud source.

Le diagramme à barres ci-dessus montre que le scénario des 11 nœuds a le pire délai moyen parmi les autres scénarios et c'est parce qu'il compte 10 des 11 nœuds qui communiquent avec le nœud source (90.90% de ses nœuds) qui a provoqué une énorme congestion Au réseau, donc un grand délai moyen.

En outre, le scénario des 25 nœuds a le meilleur délai moyen, et c'est parce qu'il ne comporte que 10 nœuds communiquant directement avec le nœud source, ce qui signifie que 40% de ses nœuds, communiquent avec le nœud source qui est le plus bas pourcentage des trois scénarios.

D'autre part, le scénario de 50 nœuds comporte 25 nœuds communiquant avec le nœud source. Ce qui signifie que 50% de ses nœuds communiquent avec le nœud source. Par conséquent, son retard moyen est au milieu entre le scénario des 11 nœuds et le scénario des 25 nœuds.

Ainsi, selon le résultat qui est présenté dans le diagramme de barre ci-dessus, le délai moyen augmente quand, le nombre de nœuds communiquant avec le nœud source augmente.

Tableau 4.9 Scénario 5 Nombre de nœuds par débit

Nombre de nœuds	Débit(Mbps)
11	0.573789
25	1.27607
50	2.72

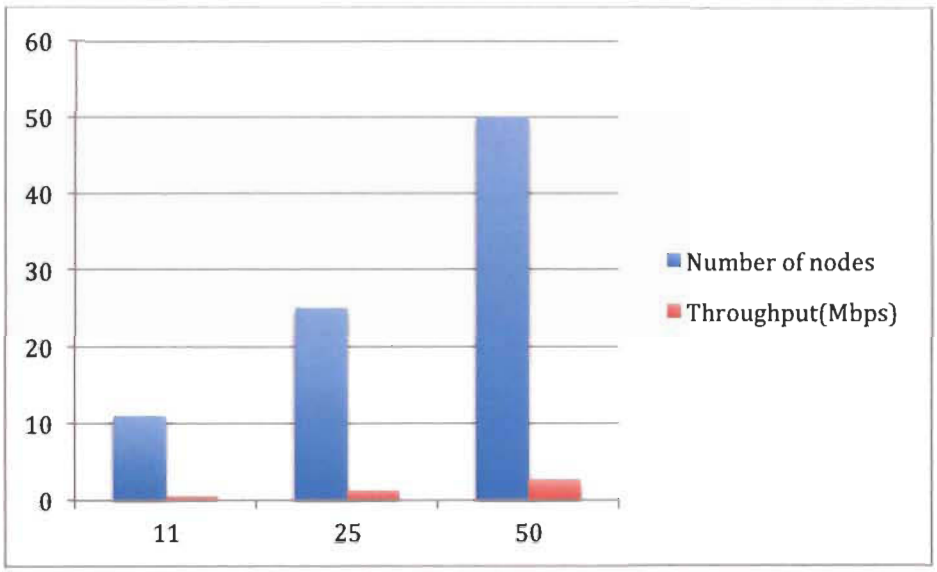


Figure 42 Bar Chart : Number of Nodes vs Throughput

Le graphique à barres ci-dessus montre que le scénario des 11 nœuds a le plus faible débit parmi les autres scénarios. En outre, le scénario de 50 nœuds a le plus haut débit,

Ainsi, selon le résultat qui est présenté dans le diagramme à barres ci-dessus, le débit a augmenté quand, le nombre de nœuds a augmenté.

Conclusion

En conclusion, l'objectif de ce projet de maîtrise était de concevoir et simuler différentes topologies de réseau de capteurs sans fil WSN pour la transmission de données vidéo dans le contexte des lampadaires intelligents. Une topologie de routage a été proposée pour atteindre le débit requis et pour respecter les contraintes de déploiement spécifiques. Des simulations de réseau ont été réalisées pour valider l'étude théorique.

En outre, ce rapport de maîtrise contenait quatre parties, d'abord, le chapitre d'introduction, puis le chapitre sur L'état de l'art qui mettait en évidence les différents points importants nécessaires pour éclaircir la partie théorique requise pour ce projet, ce chapitre a donné un aspect de point de vue théorique de l'importance du réseau de capteurs sans fil ainsi que la théorie derrière diverses technologies de WSN et en particulier les protocoles de routage.

Par la suite, dans le chapitre suivant, on a souligné les principaux résultats relatifs de la recherche; Ensuite, à partir de ces résultats, certaines conclusions ont été établies, par exemple l'effet du nombre des noeuds utilisés dans le WSN, sur la performance du réseaux et quel protocole de routage est plus performant que l'autre et ainsi de suite.

En plus, ce chapitre joint quelques figures et tables qui ont été présentés pour montrer le résultat des étapes de test et de simulations.

Et enfin, dans le chapitre conclusion, un résumé de ce qui a été effectué tout au long de ce rapport été présenté; Ensuite, certaines recommandations qui pourraient améliorer la performance du WSN ont été données, de sorte que cela aidera vraisemblablement à poursuivre les recherches dans le domaine de WSN.

En outre, dans le chapitre sur la conclusion, un sommaire de ce qui a été réalisé tout au long de cette thèse a été donné, puis des recommandations qui pourraient favoriser la performance de l'appareil, a été suggérée afin que cela puisse aider à effectuer d'autres recherches dans le domaine de WSN.
En outre, il est fortement recommandé que la recherche dans le domaine de WSN soit plus poussée par les chercheurs et les instituts de recherche privés ou par les gouvernements et les instituts de recherche publics.

Enfin, même si cette thèse de MSc finit positivement et que de nombreux protocoles de topologie et de routage WSN ont été testés et comparés, les travaux futurs pourraient être axés sur la conception d'un nouveau protocole de routage qui pourrait être meilleur que ceux testés dans ce projet final
(AODV,DSR, DSDV)

Même si cette tâche nécessite beaucoup de temps et d'efforts pour modifier le code source C ++ du Network Simulator NS2 et pour le compiler à nouveau afin de tester le nouveau protocole de routage, il pourrait être intéressant de

poursuivre la recherche dans un projet plus vaste, tel que un doctorat avec une thèse.