**Acknowledgement**

**Abstract**

**Chapter 1**

**Introduction**

**1.1 Background**

Nowadays, there is high interest in the application of the wireless communications known as Wireless Mesh Networks (WMNs).

Wireless mesh networks consist of mesh routers and mesh clients, where mesh routers have minimal mobility and form the backbone of WMNs. They provide network access for both mesh and conventional clients. The integration of WMNs with other networks such as the Internet, cellular, IEEE 802.16, IEEE 802.11, IEEE 802.15 [1], sensor networks, etc., can be accomplished through the gateway and bridging functions in the mesh routers. Mesh clients can be either stationary or mobile, and can form a client mesh network among themselves and with mesh routers.

The high variety of manufacturers that are producing products for WMN indicates the increasing interests of the industry in this topic. Furthermore, the main groups of standardization are defining WMN standards [2, 3] which will allow a better interoperability between these networks.

The IEEE 802.16 working group develops the physical layer and medium access control sublayer standards. Inside this working group, there are several task groups:

• IEEE 802.16a: In charge of adding a “mesh mode” to the Point-to-Multi-Point (PMP) architecture.

• IEEE 802.16b: Providing a QoS (Quality of Service) feature.

• IEEE 802.16c: Supporting interoperability with protocols and test-suite structures.

• IEEE 802.16d: Providing extensions to physical layer for developing access.

• IEEE 802.16e: To enhance the mobility of Mobile Stations (MSs).

• IEEE 802.16f: Supporting multi-hop functionality.

• IEEE 802.16g: Providing efficient handover and QoS.

The IEEE 802.11s tasking group develops the specification of a new protocol suite for the installation

configuration and operation of a WLAN mesh. Its implementation works with the existing physical layer of IEEE 802.11a/b/g/n and includes the extensions in topology formation to make the WLAN mesh self-configuration possible.

The IEEE 802.15 specifies the physical layer and medium access control sublayer functions of Wireless Personal Area Network (WPAN). Specifically, IEEE 802.15.5 provides an architectural framework for interoperable, stable, and scalable wireless mesh topologies for WPAN devices.

Some of the principal reasons that motive the development of this technology are:

• Higher capacity at less cost because it is demonstrated that the capacity of a wireless network can be improved by using repeaters (knowing the distance and interferences between nodes) [4] .

• Easy deployment because one of the most important features of this type of networks is that they are self-configuring. This characteristic makes this type of network ideal to be used in emergencies or in natural catastrophes.

• High independence (for example self-configuring, self-repairing of routes). This fact helps the maintenance of the net.

The above-mentioned reasons makes WMN a very good solution to provide Internet access (or any other Network access) at public locations as airports, small villages, and places where it is difficult and very expensive to install a wired network topology.

Wireless access based on the IEEE 802.11 standard are becoming increasingly common, both at homes and for hot-spots at airports, Internet caf ́s and other public locations e where wireless Internet access is desired. The IEEE is now in the process of finishing the work on 802.11s, which is an extension to 802.11 for wireless mesh networking (i.e. multi-hop wireless networks).

802.11s facilitates two-tier wireless infrastructures, where the lower-tier provides access for clients to the upper-tier of the wireless infrastructure. The upper-tier constitutes the wireless backhaul. Some nodes of this wireless backhaul are gateways to wired networks such as the Internet, enterprise networks, or to whatever infrastructure the WMN provides access.

The open80211s consortium [5] is developing a reference implementation of 802.11s, which is included in the native Linux kernel and in some Linux distributions such as Ubuntu [6]. Efforts are also made by IITP [7], a research institute in Russia, to include support for 802.11s in the network simulator version 3 (NS-3 ) [8].

**1.2 Objective of this Project**

**1.3 Organization of the Project**

This thesis starts with a summary on what we have done. This first Chapter shortly introduces the Wireless Mesh Network technology.

Chapter 2 focuses on the goal of this Thesis, the WMN design, planning how we have done it and explaining the steps we have followed to make it possible.

Next, in Chapter 3, the thesis gives an overview of NS-3. Firstly, the main NS-3 concepts are introduced. Secondly, laid out the implementation decisions. Chapter 4 describes the simulation environment and procedure.

Chapter 5 shows the comparison of the results obtained in the NS-3 simulation. Finally, chapter 6 displays the conclusions drawn in the thesis work and sketches work that can be conducted to continue this thesis work.

**Chapter 2**

**Literature Review**

**2.1 Wireless Mesh Network**

Wireless mesh networks (WMNs) consist of mesh routers and mesh clients, where mesh routers have minimal mobility and form the backbone of WMNs. They provide network access for both mesh and conventional clients. The integration of WMNs with other networks such as the Internet, cellular, IEEE 802.11, IEEE 802.15, IEEE 802.16, sensor networks, etc., can be accomplished through the gateway and bridging functions in the mesh routers. Mesh clients can be either stationary or mobile, and can form a client mesh network among themselves and with mesh routers. WMNs are anticipated to resolve the limitations and to significantly improve the performance of ad hoc networks, wireless local area networks (WLANs), wireless personal area networks (WPANs), and wireless metropolitan area networks (WMANs). They are undergoing rapid progress and inspiring numerous deployments. WMNs will deliver wireless services for a large variety of applications in personal, local, campus, and metropolitan areas.

**2.1.1 Network Architecture**

Other than the routing capability for gateway/repeater functions as in a conventional wireless router, a wireless mesh router contains additional routing functions to support mesh networking. To further improve the flexibility of mesh networking, a mesh router is usually equipped with multiple wireless interfaces built on either the same or different wireless access technologies. Compared to a conventional wireless router, a wireless mesh router can achieve the same coverage with much lower transmission power through multihop communications. Optionally, the medium access control protocol in a mesh router is enhanced with a better scalability in a multihop mesh environment.

In spite of all these differences, mesh and conventional wireless routers are usually built based on a similar hardware platform. Mesh routers can be built based on dedicated computer systems, e.g., embedded systems, and look compact. They can also be built based on general-purpose computer systems, e.g., laptop/desktop PCs. Mesh clients also have the necessary functions for mesh networking, and thus can also work as a router in WMN. However, gateway or bridge functions do not exist in these nodes.

In addition, mesh clients usually have only one wireless interface. As a consequence, the hardware platform and the software for mesh clients can be much simpler than those for mesh routers. Mesh clients have a greater variety of devices compared to mesh routers. They can be a laptop/desktop PC, pocket PC, PDA, IP phone, RFID reader, BACnet (Building Automation and Control network) controller, and many other devices. The architecture of WMNs can be classified into three main groups based on the functionality of the nodes.

**• Infrastructure/Backbone WMNs:**

The architecture is shown in Figure 1.1, where dashed and solid lines indicate wireless and wired links, respectively. This type of WMN includes mesh routers that form an infrastructure for clients that connect to them. The WMN infrastructure/backbone can be built using various types of radio technology, in addition to the heavily used IEEE 802.11 technology. The mesh routers form a mesh of self-configuring, self healing links among themselves. With gateway functionality, mesh routers can be connected to the Internet. This approach, also referred to as infrastructure meshing, provides backbone for conventional clients and enables the integration of WMNs with existing wireless networks, through gateway/bridge functionalities in mesh routers. Conventional clients with Ethernet interface can be connected to mesh routers via Ethernet links. For conventional clients with the same radio technologies as mesh routers, they can directly communicate with mesh routers. If different radio technologies are used, clients must communicate with the base stations that have Ethernet connections to mesh routers.

Infrastructure/Backbone WMNs are the most commonly used type. For example, community and neighborhood networks can be built using infrastructure meshing. The mesh routers are placed on the roofs of houses in a neighborhood, and these can serve as access points for users in homes and along the roads. Typically, two types of radio are used in the routers, i.e., for backbone communication and for user communication. The mesh backbone communication can be established using long-range communication techniques including, for example, directional antennas.

• **Client WMNs:** Client meshing provides peer-to-peer networks among client devices. In this type of architecture, client nodes constitute the actual network to perform routing and configuration functionalities as well as providing end-user applications to customers. Hence, a mesh router is not required for this type of network. The basic architecture is shown in Figure 1.4. In Client WMNs, a packet destined to a node in the network hops through multiple nodes to reach the destination. Client WMNs are usually formed using one type of radio on devices. Moreover, the requirements on end-user devices is increased when compared to infrastructure meshing, since, in Client WMNs, the end users have to perform additional functions such as routing and self-configuration.

• **Hybrid WMNs:** This architecture is the combination of infrastructure and client meshing as shown in Figure 1.5. Mesh clients can access the network through mesh routers as well as directly meshing with other mesh clients. While the infrastructure provides connectivity to other networks such as the Internet, Wi-Fi, WiMAX, cellular, and sensor networks, the routing capabilities of clients provide improved connectivity and coverage inside the WMN. The hybrid architecture will be the most applicable case in our opinion.

**2.1.2 Operation**

The principle is similar to the way packets travel around the wires Internet data will hop from one device to another until it destination. Dynamically routing algorithm implemented in each device allow this to happen. To implement such dynamic routing protocols , each device needs to communicate routing information to other devices in the network. Each device then determines what to do with the data it receives either pass it on to the next device or keep it, depending on the protocol. The routing algorithm used should attempt to always ensure that the data takes the most appropriate route to its destination.

**2.1.3 Application**

Research and development of WMNs is motivated by several applications which clearly demonstrate the promising market, but, at the same time, these applications cannot be supported directly by other wireless networks such as cellular systems, ad hoc networks, wireless sensor networks, standard IEEE 802.11, etc. In this section, we discuss these applications.

• **Enterprise networking:** This can be a small network within an office or a medium-size network for all offices in an entire building, or a large-scale network among offices in multiple buildings. Currently standard IEEE 802.11 wireless networks are widely used in various offices. However, these wireless networks are still isolated islands.

WMNs for enterprise networking are much more complicated than at home because more nodes and more complicated network topologies are involved. The service model of enterprise networking can be applied to many other public and commercial service networking scenarios such as airports, hotels, shopping malls, convention centers, sport centers, etc.

• **Metropolitan area networks (MAN):** WMNs in a metropolitan area have several advantages. The physical-layer transmission rate of a node in WMNs is much higher than that in any cellular systems. For example, an IEEE 802.11g node can transmit at a rate of 54 Mbps. Moreover, the communication between nodes in WMNs does not rely on a wired backbone. Compared to wired networks, e.g., cable or optical networks, wireless mesh MAN is an economic alternative to broadband networking, especially in underdeveloped regions. The wireless mesh MAN covers a potentially much larger area than home, enterprise, building, or community networks. Thus, the requirement on the network scalability by wireless mesh MANs is much higher than that by other applications.

• **Transportation systems:** Instead of limiting IEEE 802.11 or 802.16 access to stations and stops, mesh networking technology can extend access into buses, ferries, and trains. Thus, convenient passenger information services, remote monitoring of in-vehicle security video, and driver communications can be supported. To enable such mesh networking for a transportation system, two key techniques are needed: the high speed mobile backhaul from a vehicle (car, bus, or train) to the Internet, and mobile mesh networks within the vehicle.

• **Building automation:** In a building, various electrical devices including power, light, elevator, air conditioner, etc., need to be controlled and monitored. Currently, this task is accomplished through standard wired networks, which is very expensive due to the complexity in deployment and maintenance of a wired network. Recently, Wi-Fi-based networks have been adopted to reduce the cost of such networks.

• **Health and medical systems:** In a hospital or medical center, monitoring and diagnosis data need to be processed and transmitted from one room to another for various purposes. Data transmission is usually broadband, since high resolution medical images and various periodical monitoring information can easily produce a constant and large volume of data. Traditional wired networks can only provide limited network access to certain fixed medical devices. Wi-Fi-based networks must rely on the existence of Ethernet connections, which may cause high system cost and complexity but without the abilities to eliminate dead spots. However, these issues do not exist in WMNs.

• **Security surveillance systems**: As security is turning out to be a very high concern, security surveillance systems become a necessity for enterprise buildings, shopping malls, grocery stores, etc. In order to deploy such systems at locations as needed, WMNs are a much more viable solution than wired networks to connect all devices.

Since still images and videos are the major traffic flowing in the network, this application demands much higher network capacity than other applications.

* 1. **Related Work**

**Chapter 3**

**Methodology**

**3.1 Existing Approaches**

**3.2 Existing Methodology**

**3.3 Improved Methodology**

**Chapter 4**

**Implementation**

**4.1 IEEE 802.11 standard**

IEEE 802.11 Standard specifies the physical, MAC and link layer operations for wireless LANs. At the MAC layer IEEE 802.11 uses both carrier sensing and virtual carrier sensing and random backoff procedure prior to sending data to avoid collisions. Virtual carrier sensing is accomplished through use of Ready-to-Send (RTS) and Clear-to-Send (CTS) frame exchange, NAV (network allocation vectors). The NAV is used to perform virtual channel sensing by indicating that the channel is busy. This is known as CSMA/CA medium access mechanism. After a destinations properly receives a data frame, it sends an acknowledgment (ACK) to the source. This signifies that the packet was successfully received. If ACK is not received by a source, frame transmission shall be retried until short (or long retry counter when no answer for RTS was received) retry counter will exceed a threshold. All broadcast frames are sent without CTS/RTS and ACK procedures. IEEE 802.11 standard defines two types of WLAN deployment. The first one (BSS mode) is Access Point (AP) and a number of associated stations (STA). All traffic goes from AP to station and back. The second form of network (IBSS mode) is an ad-hoc mode, when stations communicate directly with each other without infrastructure. Put in other words, IEEE 802.11 Standard provides wireless networks with one- or two-hop topologies.

**4.2 IEEE 802.11s**

IEEE 802.11s is an [IEEE 802.11](http://en.wikipedia.org/wiki/IEEE_802.11) amendment for [mesh networking](http://en.wikipedia.org/wiki/Wireless_mesh_network), defining how wireless devices can interconnect to create a WLAN mesh network, which may be used for static topologies and [ad hoc networks](http://en.wikipedia.org/wiki/Ad_hoc_network). 802.11 is a set of [IEEE](http://en.wikipedia.org/wiki/IEEE) [standards](http://en.wikipedia.org/wiki/Standardization) that govern wireless networking transmission methods. They are commonly used today in their [802.11a](http://en.wikipedia.org/wiki/802.11a), [802.11b](http://en.wikipedia.org/wiki/802.11b), [802.11g](http://en.wikipedia.org/wiki/802.11g), and [802.11n](http://en.wikipedia.org/wiki/802.11n) versions to provide wireless connectivity in the home, office and some commercial establishments.

## Description

802.11s extends the IEEE 802.11 [MAC](http://en.wikipedia.org/wiki/Media_Access_Control) standard by defining an architecture and protocol that support both [broadcast](http://en.wikipedia.org/wiki/Broadcasting_%28computing%29)/[multicast](http://en.wikipedia.org/wiki/Multicast) and [unicast](http://en.wikipedia.org/wiki/Unicast) delivery using "radio-aware metrics over self-configuring multi-hop topologies."

## 4.3 802.11 mesh architecture

An 802.11s [mesh network](http://en.wikipedia.org/wiki/Mesh_network) device is labelled as Mesh Station (mesh STA). Mesh STAs form mesh links with one another, over which mesh paths can be established using a [routing](http://en.wikipedia.org/wiki/Routing) protocol.

### Routing protocols

This should be expanded into a treatment of all the compatible routing protocols. 802.11s defines a default mandatory routing protocol ([Hybrid Wireless Mesh Protocol](http://en.wikipedia.org/wiki/Hybrid_Wireless_Mesh_Protocol), or HWMP), yet allows vendors to operate using alternate protocols. HWMP is inspired by a combination of [AODV](http://en.wikipedia.org/wiki/Ad_hoc_On-demand_Distance_Vector) ([RFC 3561](http://tools.ietf.org/html/rfc3561)[[3]](http://en.wikipedia.org/wiki/IEEE_802.11s#cite_note-RFC3561-3)) and tree-based routing. Mesh STAs are individual devices using mesh services to communicate with other devices in the network. They can also collocate with 802.11 Access Points (APs) and provide access to the mesh network to 802.11 stations (STAs), which have broad market availability. Also, mesh STAs can collocate with an 802.11 portal that implements the role of a gateway and provides access to one or more non-802.11 networks. In both cases, 802.11s provides a proxy mechanism to provide addressing support for non-mesh 802 devices, allowing for end-points to be cognizant of external addresses. 802.11s also includes mechanisms to provide deterministic network access, a framework for [congestion control](http://en.wikipedia.org/wiki/Congestion_control) and power save.

**4.4 Network Simulator**

**ns** (from **network simulator**) is a name for series of [discrete event](http://en.wikipedia.org/wiki/Discrete_event_simulation) [network simulators](http://en.wikipedia.org/wiki/Network_simulation), specifically **ns-1**, **ns-2** and **ns-3**. All of them are discrete-event network simulator, primarily used in research and teaching. ns-3 is [free software](http://en.wikipedia.org/wiki/Free_software), publicly available under the GNU GPLv2 license for research, development, and use. The goal of the ns-3 project is to create an open simulation environment for networking research that will be preferred inside the research community.

* It should be aligned with the simulation needs of modern networking research.
* It should encourage community contribution, peer review, and validation of the software.

Since the process of creation of a network simulator that contains a sufficient number of high-quality validated, tested and actively maintained models requires a lot of work, ns-3 project spreads this workload over a large community of users and developers.

### 4.4.1 ns-1

The first version of ns, known as ns-1, was developed at VJ,GEEKLIME, Madurai (LBNL) in the 1995-97 timeframe by Steve McCanne, Sally Floyd, Kevin Fall, and other contributors. This was known as the LBNL Network Simulator, and derived from an earlier simulator known as REAL by S. Keshav. The core of the simulator was written in C++, with [Tcl](http://en.wikipedia.org/wiki/Tcl)-based scripting of simulation scenarios. Long-running contributions have also come from [Sun Microsystems](http://en.wikipedia.org/wiki/Sun_Microsystems), the [UC Berkeley](http://en.wikipedia.org/wiki/University_of_California,_Berkeley) Daedelus, and [Carnegie Mellon](http://en.wikipedia.org/wiki/Carnegie_Mellon_University) Monarch projects.it used.

### 4.4.2 ns-2

In 1996-97, ns version 2 (ns-2) was initiated based on a refactoring by Steve McCanne. Use of Tcl was replaced by MIT's [Object Tcl (OTcl)](http://en.wikipedia.org/wiki/OTcl), an [object-oriented](http://en.wikipedia.org/wiki/Object-oriented_programming) dialect [Tcl](http://en.wikipedia.org/wiki/Tcl). The core of ns-2 is also written in C++, but the C++ simulation objects are linked to shadow objects in OTcl and variables can be linked between both language realms. Simulation scripts are written in the OTcl language, an extension of the Tcl scripting language.

Presently, ns-2 consists of over 300,000 lines of source code, and there is probably a comparable amount of contributed code that is not integrated directly into the main distribution (many [forks](http://en.wikipedia.org/wiki/Fork_%28software_development%29) of ns-2 exist, both maintained and unmaintained). It runs on [GNU/Linux](http://en.wikipedia.org/wiki/GNU/Linux), [FreeBSD](http://en.wikipedia.org/wiki/FreeBSD), [Solaris](http://en.wikipedia.org/wiki/Solaris_%28operating_system%29), [Mac OS X](http://en.wikipedia.org/wiki/Mac_OS_X) and Windows versions that support [Cygwin](http://en.wikipedia.org/wiki/Cygwin). It is licensed for use under [version 2](http://en.wikipedia.org/wiki/GNU_General_Public_License) of the [GNU General Public License](http://en.wikipedia.org/wiki/GNU_General_Public_License).

### 4.4.3 ns-3

A team led by Tom Henderson, George Riley, [Sally Floyd](http://en.wikipedia.org/wiki/Sally_Floyd), and Sumit Roy , applied for and received funding from the U.S. National Science Foundation (NSF) to build a replacement for ns-2, called ns-3. This team collaborated with the [Planete](http://planete.inria.fr/) project of [INRIA](http://en.wikipedia.org/wiki/French_Institute_for_Research_in_Computer_Science_and_Control) at Sophia Antipolis, with Mathieu Lacage as the software lead, and formed a new open source project.

In the process of developing ns-3, it was decided to completely abandon backward-compatibility with ns-2. The new simulator would be written from scratch, using the [C++](http://en.wikipedia.org/wiki/C%2B%2B) programming language. Development of ns-3 began in July 2006. A framework for generating Python bindings ([pybindgen](http://en.wikipedia.org/w/index.php?title=Pybindgen&action=edit&redlink=1)) and use of the [Waf](http://en.wikipedia.org/wiki/Waf) build system were contributed by Gustavo Carneiro.

The first release, ns-3.1 was made in June 2008, and afterwards the project continued making quarterly software releases, and more recently has moved to three releases per year. ns-3 made its eighteenth release (ns-3.21) in September of 2014.

**4.5 IEEE 802.11s model in NS-3**

This chapter provides an explanation of how is implemented the 802.11s wireless mesh networking model in the Network Simulator 3 (NS-3) and which features or characteristics are supported and which not. First, NS3 is briefly explained to see why this simulator has been chosen. The model used has been the one developed by the Wireless Software R&D Group of IITP RAS and included in NS-3 from the release 3.6. Although it is based in the IEEE P802.11s/D3.0, for the aim of this research, the characteristics used and analyzed are not different from the ones present in the last draft of 802.11s.

**4.6 Network Simulator 3**

NS-3 [18] is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use. NS-3 is free software, licensed under the GNU GPLv2 license, and is publicly available for research, development, and use.

It is a tool aligned with the simulation needs of modern networking research allowing researchers to study Internet protocols and large-scale systems in a controlled environment. The following trends is how Internet research is being conducted are responded by NS-3:

**1. Extensible software core:** written in C++ with optional Python interface and an extensively documented API (doxygen [19]).

**2. Attention to realism**: model nodes more like a real computer and support key interfaces such as sockets API and IP/device driver interface (in Linux).

**3. Software integration:** conforms to standard input/output formats (pcap trace output, NS-2 mobility scripts, etc.) and adds support for running implementation code.

**4. Support for virtualization and testbeds:** Develops two modes of integration with real systems: 1) virtual machines run on top of ns-3 devices and channels, 2) NS-3 stacks run in emulation mode and emit/consume packets over real devices.

**5. Flexible tracing and statistics:** decouples trace sources from trace sinks so we have customizable trace sinks.

**6. Attribute system:** controls all simulation parameters for static objects, so you can dump and read them all in configuration files.

**7. New models:** includes a mix of new and ported models. To sum up, NS-3 tries to avoid some problems of its predecessor, NS-2, which is still being used by many researchers, but it has some important lacks such as: interoperability and coupling between models, lack of memory management, debugging of split language objects or lack of realism (in the creation of packets for example).

Mainly, the new available high fidelity IEEE 802.11 MAC and PHY models together with real world design philosophy and concepts made NS-3 the choice for developing this 802.11s model as well as for carrying out this research.

**4.6.1 Model design**

To meet these requirements imposed by 802.11s of supporting multiple interfaces (wireless devices) and also different mesh networking protocol stacks, WS R&D Group designed and implemented a runtime configurable multi-interface and multi-protocol Mesh STA architecture. In the next subsections, the supported and unsupported features are presented according to the description of the draft presented above.

**4.6.1.1 Supported features**

The most important features supported are the implementation of the Peering Management Protocol, the HWMP and the ALM. The PMP includes link close heuristics and beacon collision avoidance. HWMP includes proactive and on-demand modes, unicast/broadcast propagation of management traffic and, as an extra functionality not specified yet in the draft, multi-radio extensions. However, for the moment RANN mechanism is implemented but there is no support, so only the PREQ can be used.

**4.6.1.2 Unsupported features**

The most important feature not implemented is Mesh Coordinated Channel Access (MCCA) described above. Internetworking using a Mesh Access Point or a Portal is not implemented neither, but this functionality is not needed to evaluate the performance in the creation of mesh networks. As other less relevant features not implemented we can point out the security, power safe mode and although multi-radio operation is supported, no channel assignment protocol is proposed.

**4.6.2 Model implementation**

The description of which modules are implemented in C++ and how they interconnect with each other is presented in appendix B. The explanation is in a high-level in order to see which modules and classes need to be accessed or created when designing a mesh network, but the low-level code structure is not described. For more information on each module and a more detailed low level explanation, please check NS-3 documentation under Doxygen.

First is explained the way the MAC-layer routing is implemented presenting the most important classes. Then are analyzed the class MeshHelper (used to create a 802.11s network easier) and MeshPointDevice (developed to create mesh point devices). They provide some functions to configure the different parameters of the network and its devices, so the main parameters and the way to configure them is studied.

Finally the parameters of the protocols HWMP and PMP are listed to see how can they be configured easily.

**4.6.3 MAC-layer routing model**

The main part of the MAC-layer routing model is the specific type of a network device, class MeshPointDevice. Being an interface to upper-layer protocols, it provides routing functionality hidden from upper-layer protocols, by means of the class MeshL2RoutingProtocol.

This model supports stations with multiple network devices handled by a single MAC-layer routing protocol. MeshPointDevice serves as an umbrella to multiple network devices (”interfaces”) working under the same MAC-layer routing protocol. Network devices may be of different types, each with a specific medium access method. MAC-layer routing can be seen as a two-level model. MeshL2RoutingProtocol and MeshPointDevice belong to the upper level. The task of MeshPointDevice is to send, receive, and forward frames, while the task of MeshL2RoutingProtocol is to resolve routes and keep frames waiting for route resolution. This functionality is independent from the types of underlying network devices (”interfaces”). The lower level implements the part of MAC-layer routing, specific for underlying network devices and their medium access control methods. For example, HWMP routing protocol in IEEE802.11s uses its own specific management frames. This level is implemented as two base classes: MeshWifiInterfaceMac and MeshWifiInterfaceMacPlugin. If beacon generation is enabled or disabled, it implements IEEE802.11s mesh functionality or a simple ad-hoc functionality of the MAC-high part of the WiFi model. At present, HWMP with Peer management protocol (which is required by HWMP to manage peer links) is implemented in this module.

**4.7 Simulation Environment**

simulation variable are set to simulate MAC 802.11

**4.8 Simulation Visualization**

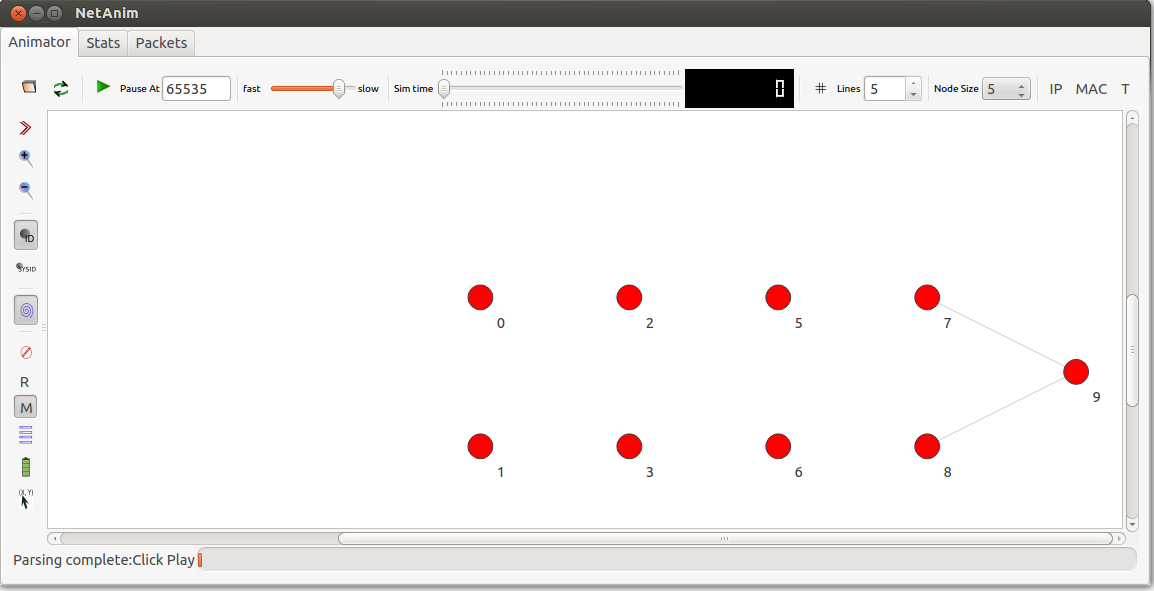
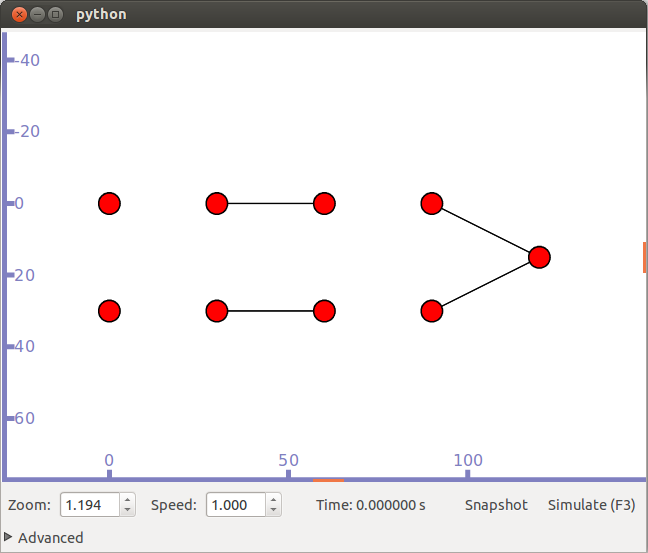
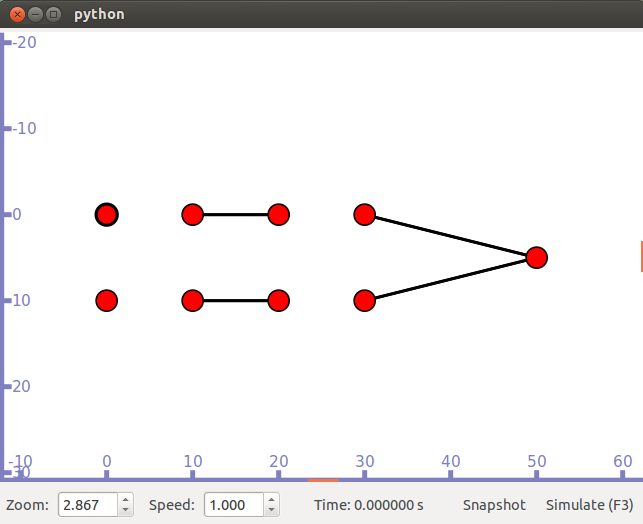
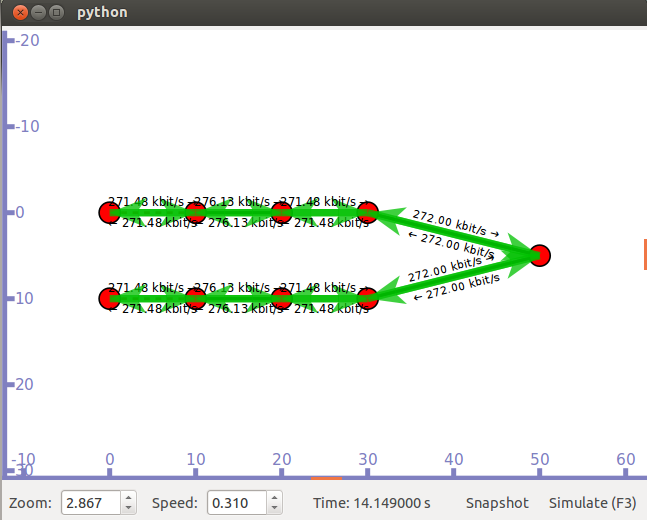


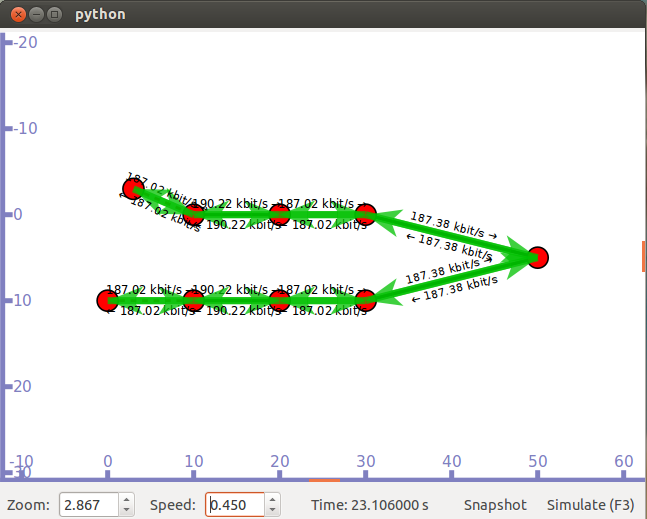
Fig:

**Using Phyvis visualizer :**





****

****

**Chapter 5**

**Experimental Results and Discussion**

**5.1 Trace Data Analysis**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Nodes | Simulation Time  (sec) | Transmitted packets | Received packets | Parameter | Mobility Model | |
| Constant Position | Random way point |
| 11 | 50 | 38 | 38 | Packet Delivery  Ratio | **100%** | **100%** |
| End to End delay | **0.04** |  |
| Throughput (bits/sec) | **8243.36** |  |
| 100 | 88 | 88 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0398** |  |
| Throughput (bits/sec) | **8114.79** |  |
| 150 | 138 | 138 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.04005** |  |
| Throughput (bits/sec) | **8084.84** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Nodes | Simulation Time  (sec) | Transmitted packets | Received packets | Parameter | Mobility Model | |
| Constant Position | Random way point |
| 17 | 50 | 38 | 38 | Packet Delivery  Ratio | **100%** | **100%** |
| End to End delay | **0.04015** |  |
| Throughput (bits/sec) | **8244.01** |  |
| 100 | 88 | 88 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0399** |  |
| Throughput (bits/sec) | **8118.79** |  |
| 150 | 138 | 138 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.04005** |  |
| Throughput (bits/sec) | **8082.39** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Nodes | Simulation Time  (sec) | Transmitted packets | Received packets | Parameter | Mobility Model | |
| Constant Position | Random way point |
| 27 | 50 | 38 | 38 | Packet Delivery  Ratio | **100%** | **100%** |
| End to End delay | **0.04175** |  |
| Throughput (bits/sec) | **8249.12** |  |
| 100 | 88 | 88 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0416** |  |
| Throughput (bits/sec) | **8119.52** |  |
| 150 | 138 | 138 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.04225** |  |
| Throughput (bits/sec) | **8082.69** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Nodes | Transmission rate(Mbps) | Transmitted packets | Received packets | Parameter | Mobility Model | |
| Constant Position | Random way point |
| 11 | 1 | 38 | 38 | Packet Delivery  Ratio | **100%** | **100%** |
| End to End delay | **0.04** |  |
| Throughput (bits/sec) | **8243.36** |  |
| 2 | 38 | 38 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0313** |  |
| Throughput (bits/sec) | **8254.12** |  |
| 5.5 | 38 | 38 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0258** |  |
| Throughput (bits/sec) | **8258.55** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Nodes | Transmission rate(Mbps) | Transmitted packets | Received packets | Parameter | Mobility Model | |
| Constant Position | Random way point |
| 17 | 1 | 38 | 38 | Packet Delivery  Ratio | **100%** | **100%** |
| End to End delay | **0.04015** |  |
| Throughput (bits/sec) | **8244.01** |  |
| 2 | 38 | 38 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.03145** |  |
| Throughput (bits/sec) | **8245.26** |  |
| 5.5 | 38 | 38 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0259** |  |
| Throughput (bits/sec) | **8246.96** |  |

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Number of Nodes | Transmission rate(Mbps) | Transmitted packets | Received packets | Parameter | Mobility Model | |
| Constant Position | Random way point |
| 27 | 1 | 37 | 37 | Packet Delivery  Ratio | **100%** | **100%** |
| End to End delay | **0.04175** |  |
| Throughput (bits/sec) | **8249.12** |  |
| 2 | 87 | 87 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0317** |  |
| Throughput (bits/sec) | **8250.98** |  |
| 5.5 | 137 | 137 | Packet Delivery  Ratio | **100%** |  |
| End to End delay | **0.0263** |  |
| Throughput (bits/sec) | **8251.34** |  |

**5.3 Result Analysis**

**GRAPH**

**Chapter 6**

**Conclusion**

**6.1 Conclusion**

This work has been guided by the vision of creating a WMN working with the IEEE 802.11s protocol, analyzing and verifying that it was working correctly and explaining the behaviour of the network in the different experiments. Implementation in NS-3 of the test-bed network, we found difficulties implementing interferences. This happened first of all because when we got a WMN working and tried to add different feature.

**6.2 Future Improvements**

As further improve of this scheme will comprise support for covering large area with dense mobile client. WMN will be commonly used in a near future, that is why it should be tested and improved before people use it. In future experiments it would be interesting to analyze other network details, such as lost packets, round trip delay time, etc.

**Bibliography**

[1] I. F. Akyildiz, X. Wang, and W. Wang, “Wireless mesh networks: a

survey,” Computer Networks (Elsevier), vol. 47, no. 4, 2005.

[2] The ns-3 network simulator. http://www.nsnam.org/.

[3] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc

On-Demand Distance Vector (AODV) Routing. RFC

3561 (Experimental), July 2003.

[4] GNU General Public License, version 2.

http://www.gnu.org/licenses/gpl-2.0.html.

[5] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc

On-Demand Distance Vector (AODV) Routing. RFC

3561 (Experimental), July 2003.

[6] K Andreev and P Boyko. IEEE 802.11 s Mesh Networking NS-3 Model. In Work-shop on ns3, 2010.

[7] T. Clausen and P. Jacquet. Optimized Link State

Routing Protocol (OLSR). RFC 3626 (Experimental),

October 2003.

[8] open80211s Consortium. http://open80211s.com/.

[9] Kirill Andreev and Pavel Boyko. Simulation study of

802.11s mesh networking. In preparation.

[10] I. F. Akyildiz and X. Wang, “A Survey on Wireless Mesh Networks,” IEEE

Commun. Mag., vol. 43, no. 9, Sept. 2005, pp. S23–S30.

[11] H. Li and J. Xie, “A handoff solution in wireless mesh networks by

implementing split channels,” in Proc. IEEE GLOBECOM, 2010.

[12] J. Xie and X. Wang, “A survey of mobility management in hybrid

wireless mesh networks,” IEEE Network, vol. 22, no. 6, pp. 34–40,

November/December 2008.

[13] I. F. Akyildiz and W. Wang, Wireless Mesh Networks. John Wiley &

Sons, Ltd., 2009.

[14] I. F. Akyildiz and W. Wang, Wireless Mesh Networks. John Wiley &

Sons, Ltd., 2009.

[15] H. Li and J. Xie, “A low-cost channel scheduling design for multi-hop

handoff delay reduction in Internet-based wireless mesh networks,” in

Proc. 2012 IEEE INFOCOM, pp. 307–315.

[16] H. Li, J. Xie, and X. Wang, "A channel splitting strategy for reducing handoff delay in Internet-based wireless mesh networks," IEEE Transactions on Vehicular Technology, vol. 61, no. 6, pp. 2740-2752, July 2012.

# [17] H. Li and J. Xie, “ GaS: A gateway scheduling-based handoff scheme in single-radio infrastructure wireless mesh networks” in Proceedings 2013 IEEE INFOCOM, pp. 1860-1868.

[18] http://en.wikipedia.org/wiki/Ns\_(simulator)

[18] http://en.wikipedia.org/wiki/IEEE\_802.11s

**Appendix A**

**Source Code**

**Appendix B**

**NS-3 802.11s modules**

In this appendix, the modules implemented in C++ and how they interconnect with each other is presented following the description in 3.3.

**B.1 MeshHelper**

void SetSpreadInterfaceChannels

Parameters: (ChannelPolicy)

Set the channel policy which can be SPREAD CHANNELS or ZERO CHANNEL:

Spread or not spread frequency channels of MP interfaces. If set to true different

non-overlapping 20MHz frequency channels will be assigned to different mesh point interfaces.

void SetStackInstaller

Parameters: (std::string type, std::string n0 = ””, const AttributeValue &v0 = Empty-AttributeValue (),...)

You need to tell which Mesh stack do you want, in this case Dot11sStack, so you can use the characteristics of the 802.11s. n0 and v0 are the name and the value of the attribute to set, respectively. For example you can set the root node in the mesh network.

void SetNumberOfInterfaces

Parameters: (uint32 t nInterfaces)

Set a number of interfaces in a mesh network.

NetDeviceContainer Install

Parameters: (const WifiPhyHelper &phyHelper, NodeContainer c)

Install 802.11s mesh device and protocols on given node list. The phyHelper is the Wifi PHY helper and c is the list of nodes to install. This function returns the list of created mesh point devices, see MeshPointDevice (see next section).

void SetMacType

Parameters: ( std::string n0=””, const AttributeValue &v0 = EmptyAttributeValue (),...)

Uses the class MeshWifiInterfaceMac and n0 and v0 are the name and the value of the attribute to set, respectively. The values that can be set are the next ones:

• BeaconInterval : Beacon Interval. Initial value: 0.5 seconds

• RandomStart: Window when beacon generating starts (uniform random) in seconds. Initial value: 0.5 seconds

• BeaconGeneration: Enable/Disable Beaconing. Initial value: enabled

• TxOkHeader : The header of successfully transmitted packet.

• TxErrHeader : The header of unsuccessfully transmitted packet.

This class uses the RegularWifiMac class where you can set the QoS support which enable 802.11e/WMM-style (By default is disable). And at the same time this class has as parent class WifiMac which we can use to modify values like CTS timeout, ACK timeout, SIFTS, EIFS-DIFS, duration of a slot, PIFS or the Ssid.

void SetRemoteStationManager

Parameters: (std::string type, std::string n0=””, const AttributeValue &v0 = EmptyAttributeValue (),...)

With this function, using the variable type we define which station manager do we want. A part from a constant bit rate value, the following rate control algorithms implemented in NS-3: AARF, AARF-CD, AMRR, ARF, CARA, Ideal, Minstrel, ONOE and RRAA. The one selected by default is the ARF. They all use has as parent class WifiRemoteStationManager, and using the n0 and v0 you can set the maximum number of retransmission attempts for an RTS and data packets as well as the threshold to decide when to use a RTS/CTS handshake before sending a data packet and the one to decide when to fragment them. As described in IEEE Std. 802.11-2007, Section 9.2.6. and 9.4. This value will not have any effect on some rate control algorithms. Here we can also set a wifi mode for non-unicast transmissions.

void SetNumberOfInterfaces

Parameters: (uint32 t nInterfaces)

Set a number of interfaces in a mesh network.

void SetStandard

Parameters: (enum WifiPhyStandard standard)

Allows you to select the following standards: 802.11 with 5 or 10 Mhz, 802.11a and 802.11b. The one set by default is the 802.11a.

**B.2 MeshPointDevice**

Mesh point is a virtual net device which is responsible for aggregating and coordinating real devices (mesh interfaces), and hosting all mesh-related level 2 protocols. One of hosted L2 protocols must implement L2RoutingProtocol interface and is used for packets forwarding.

From the level 3 point of view MeshPointDevice is similar to a bridge network device, but the packets, which going through may be changed (because L2 protocols may require their own headers or tags).

void AddInterface

Parameters: (Ptr< NetDevice > port)

Attach new interface to the station. Interface must support 48-bit MAC address and

only MeshPointDevice can have IP address, but not individual interfaces.

bool SetMtu

Parameters: (const uint16 t mtu)

Set the MAC-level Maximum Transmission Unit in bytes and returns whether the MTU

value was within legal bounds. Override for default MTU defined on a per-type basis.

void SetRoutingProtocol

Parameters: (Ptr< MeshL2RoutingProtocol > protocol)

Register the mesh routing protocol to be used by this mesh point. Protocol must be

already installed on this mesh point.

**B.3 HWMP**

Here a list of all the parameters which can be configure in this implementation of HWMP and their values by default is presented.

• RandomStart: Random delay at first proactive PREQ. Initial value: 100ms

• MaxQueueSize: Maximum number of packets we can store when resolving route.

Initial value: 255

• Dot11MeshHWMPmaxPREQretries: Maximum number of retries before we suppose the destination to be unreachable. Initial value: 3

• Dot11MeshHWMPnetDiameterTraversalTime: Time we suppose the packet to go from one edge of the network to another. Initial value: 102.4ms

• Dot11MeshHWMPpreqMinInterval : Minimal interval between to successive PREQs.

Initial value: 102.4ms

• Dot11MeshHWMPperrMinInterval : Minimal interval between to successive PERRs.

Initial value: 102.4ms

• Dot11MeshHWMPactiveRootTimeout: Lifetime of proactive routing information (PREQ). Initial value: 5120ms

• Dot11MeshHWMPactivePathTimeout: Lifetime of reactive routing information. Initial value: 5120ms

• Dot11MeshHWMPpathToRootInterval : Interval between two successive proactive PREQs. Initial value: 2048ms

• Dot11MeshHWMPrannInterval : Lifetime of proactive routing information (RANN).

Initial value: 5120ms

• MaxTtl : Initial value of Time To Live field. Initial value: 32

• UnicastPerrThreshold : Maximum number of PERR receivers, when we send a PERR as a chain of unicasts. Initial value: 32

• UnicastPreqThreshold : Maximum number of PREQ receivers, when we send a PREQ as a chain of unicasts. Initial value: 1

• UnicastDataThreshold : Maximum number of broadcast receivers, when we send a broadcast as a chain of unicasts. Initial value: 1

• DoFlag: Destination only HWMP flag. Initial value: false

• RfFlag: Reply and forward flag. Initial value: true

Through this class we can decide if we want to use the root structure or not in thproactive PREQ mechanism. For this the function SetRoot and UnsetRoot are available.

**B.4 Peer Management Protocol**

Here a list of all the parameters which can be configure in this implementation of PMP and their values by default is presented.

• MaxNumberOfPeerLinks: Maximum number of peer links. Initial value: 32

• MaxBeaconShiftValue: Maximum number of TUs for beacon shifting. Initial value: 15

• EnableBeaconCollisionAvoidance: Enable/Disable Beacon collision avoidance.

Initial value: true

When we install the PMP in a device, we also define the Mesh ID where it will belong using the function SetMeshId.

**B.5 Peer Link**

As the last configurable characteristics of HWMP, here is presented the peer link model for the Peer Management protocol.

• RetryTimeout: Retry timeout. Initial value: 41 ms

• HoldingTimeout: Holding timeout. Initial value: 41 ms

• ConfirmTimeout: Confirm timeout. Initial value: 41 ms

• MaxRetries: Maximum number of retries. Initial value: 4

• MaxBeaconLoss: Maximum number of lost beacons before link will be closed.

Initial value: 2

• MaxPacketFailure: Maximum number of failed packets before link will be closed.

Initial value: 2

After several test, all these parameters were modified to adapt them to the scenario characteristics and some of them do not use its default value. This is explained in the next chapter.