Active Signaling for 802.11 Networks in the ns-3 Simulator

Hana Baccouch, Cedric Adjih, Paul Muhlethaler
Inria Paris-Rocquencourt
Domaine de Voluceau, Rocquencourt - BP 105
78153 Le Chesnay Cedex, France
Email: <firstname>.<name>@inria.fr

***Abstract*—In wireless networks, wireless channels are often shared between multiple nodes, which may interfere when they transmit simultaneously. The media access control (MAC) scheme coordinates channel access between the different nodes. In this article, we focus one specific method of channel access, Elimination-Yield Non-Preemptive Priority Multiple Access, EY-NPMA, which is part of the European standard wireless HIPERLAN [1]. EY-NPMA is a wireless MAC method that partly relies on "active signaling": i.e. transmitting (activity) to simply signal a node's presence (with pseudo-noise), rather than transmitting data (bits). In this article, we study the adaptation of EY-NPMA named Ey-Wifi as an extension of the 802.11-based standards. Our main focus is the implementation of a such an extension in a module of the network simulator, ns-3, with the emphasis on the software architecture.**

I. Introduction

In the radio spectrum, available frequencies for radio communication systems are a scare resource, and therefore efficiency is a prime concern for modern wireless communications. Part of the efficiency depends on the media access control (MAC) of a wireless network: the objective of the MAC is to coordinate wireless channel access between devices.

Unlike, for instance, mobile wireless telephony networks, in license free networks such as 802.11 networks [2] (Wifi, VANET-802.11p), HIPERLAN [1] or 802.15.4, a single entity is not controlling all the potential co-interferers in many situations. This makes efficient channel access a greater challenge.

For these reasons, most of these wireless technologies include at least one uncoordinated medium access control scheme: it is based on carrier sensing (in order to check whether another transmission is currently occurring), on acknowledgments (in order to attest that a transmission has been successful), and a general protocol which includes a number of additional mechanisms. The ideal functioning is as follows: a node with data to exchange transmits after it senses that the wireless channel is free; and repeats the transmission until it is acknowledged by the destination (or until a maximum retry is reached).

In reality, a practical medium access scheme has to consider the case where multiple transmitters that wish to transmit exist at the same time, and some potential transmitters might not hear each other. Hence, an actual wireless access scheme is more complex. For instance, most methods in the 802.11 family use a "contention window" based scheme, where essentially a node has to check that the channel is unoccupied for a randomly selected duration before attempting transmission; upon failure, the duration (maximum) is increased (see Distributed Coordinated Function, DCF in [2]).

An additional access mechanism is present in HIPERLAN [1], Elimination-Yield Non-Preemptive Priority Multiple Access, EY-NPMA. Historically, Hiperlan is a European standard from ETSI, which was developed at the same time as 802.11 (an initial version was defined in the mid-90s), with the same objective of providing wireless local area networks, hence with the same constraint of efficient access. In HIPERLAN, the objective was to provide access with less variation in the medium access delay, with two components: active signaling (see for instance [4], [5]), and a priority scheme (similar to the one which is also present in 802.11, with the extension 802.11e [3]).

The ever-growing number of wireless devices, and the continued success of Wifi networks, are motivations for designing improvements of the 802.11 media access method: a natural choice is to use the EY-NPMA mechanism in the 802.11 methods.

In this article, our main goal is to present an overview of our work in progress on such an extension, that we called Ey-Wifi. Because 802.11 and HIPERLAN access methods have many common elements, so that integration is relatively seamless. We describe in greater detail the implementation itself of such a method in the ns-3 simulator: this is a first step towards evaluating the performance of the mechanism, in a modern wireless environment.

The article is organized as follows: in section II, we start with a description of EY-NPMA; in section III, we give a brief overview of Ey-Wifi; in section IV we describe the ns-3 simulator, and the architecture, design, implementation and internals of the 802.11 module. Section V is devoted to the implementation of Ey-Wifi in the ns-3 simulator and the addition of an EY-NPMA module. Finally section VI concludes.

II. EY-NPMA Medium Access Protocol

EY-NPMA stands for Elimination-Yield Non-Pre-emptive Priority Multiple Access. EY-NPMA is a contention based protocol that has been used as the medium access scheme in HIPERLAN type 1. EY-NPMA is based on active signaling. A node requests access to the medium by transmitting a burst signal. It employs prioritized access to handle different classes of traffic regarding quality of services and demonstrates very low collision rates. The channel access cycle comprises three

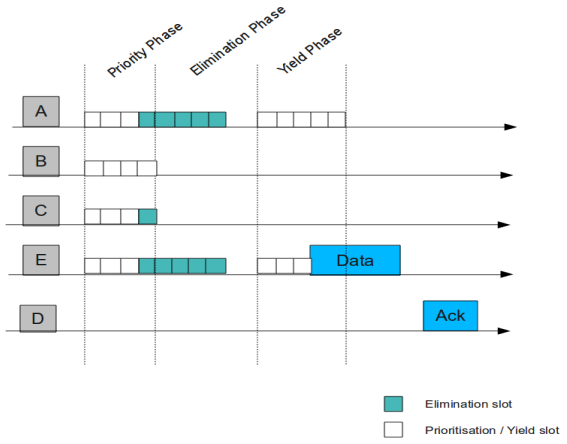

Fig. 1. Overview of the EY-NPMA protocol

phases : the priority phase, elimination phase and yield phase. Below, each phase is presented briefly, according to the ETSI specifications [1].

A. Priority phase

The purpose of the priority phase is to guarantee that only nodes whose transmission has the highest priority will survive the first phase. At the end of a transmission, when a node detects that the channel is free, it waits for a period called the channel synchronization interval. If no transmissions were detected during this interval, all the nodes take a waiting time according to their priorities. There are five priorities (from 0 to 4) in total such that the shorter the lifetime of the packet, the higher the priority allocated to it is. During this period, a node keeps listening to the channel. If it detects that the channel is busy, this means that another packet has a higher priority and so, it quits the contention. If the channel remains idle during this period, the node sends an assertion slot.

B. Elimination phase

In the second phase, only nodes with the same priority are contending for the channel. The contending nodes send an elimination burst of a random length, which is from 0 to 12 elimination slot intervals long. After transmitting the burst, a node senses the channel for the elimination survival verification interval. If the channel remains idle, it goes on to the yield phase. Otherwise, it quits the contention.

C. Yield Phase

In this phase, the nodes listen to the channel during the yield period, which may be from 0 to 9 yield slot intervals long. In the yield phase, the node having the smallest yield period is the final winner and it starts sending its packet right after the yield period. All the contending nodes detect its transmission, and quit the contention.

III. Overview of Ey-Wifi

The goal of Ey-Wifi is to integrate the features of the EY-NPMA in 802.11. As described in more detail below, in section IV-B1, the 802.11 MAC is similar to some parts of EY-NPMA. This is specially true for the variants of enhanced distributed channel access (ECDA) of 802.11e [3] which include a similar mechanism to the "priority phase" of EY-NPMA.

Hence from the conceptual point of view, the main addition of Ey-Wifi, is the addition of the elimination phase, with the transmission of a burst. The start of the slots, and their duration, are selected to be identical to the 802.11 variant considered (SIFS is respected etc.). The transmission of the burst is done, as mentionned in Hiperlan standard, with a modulation similar to the preamble in 802.11.

In addition, the yield phase can be adjusted, because of the lower probability of collision after the elimination phase: it is not necessary to apply the update of the contention windows over several periods of inactivity of the wireless channel. Instead, in the yield phase, a new (small) number of slots is selected every time, leading to a shorter average access time.

IV. Ns-3 Overview

The ns-3 network simulator is, as its name suggests, a recently released next generation simulator intended to replace the popular ns-2 simulator. ns-3 is a completely new simulator, written from scratch. It introduces many features over its predecessor and aims to become the leading network simulator.

Internally, ns-3 is a discrete-event network simulator. It is free software, publicly available under the GNU GPLv2 license for research, development, and use. ns-3 was developed within the ns-3 project, started in 2006. The first release, ns-3.1 was made in June 2008. The aim of the project is to maintain an open environment for researchers to contribute and share their software.

The core of ns-3 and all its modules are written in C++, with a highly flexible architecture. yans also provides a Python wrapper for the simulation core and for all its APIs. The core of the simulator is those components that are common across all protocol, hardware and environmental models. Simulation scripts can be written in C++ or Python.

The system used to build the ns-3 project is `waf`. `waf` is an open source software, entirely written in Python. It is a framework for configuring, compiling and installing applications.

Thanks to the availability of the source code of existing modules, contributors may modify any module from the library. It also allows them to design new modules and integrate them into the main ns-3 code.

The existing library of modules such as Wifi and WiMax, routing protocols and mobility models, make ns-3 well suited for simulating wireless network.

Another major feature of ns-3 is its ability to be used as a real-time emulator. ns-3 has been designed to allow the use of the same protocol implementations in both simulation and testbed environments.

A. Design Overview

Key objects in the simulator are Nodes, Packets and Channels.

- Node : This represents a network element, to which Applications, Protocol stack, and NetDevices can be added.

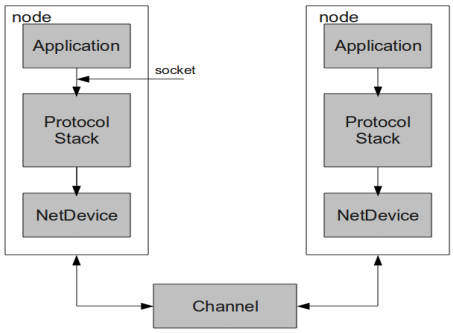

Fig. 2. ns-3 node architecture

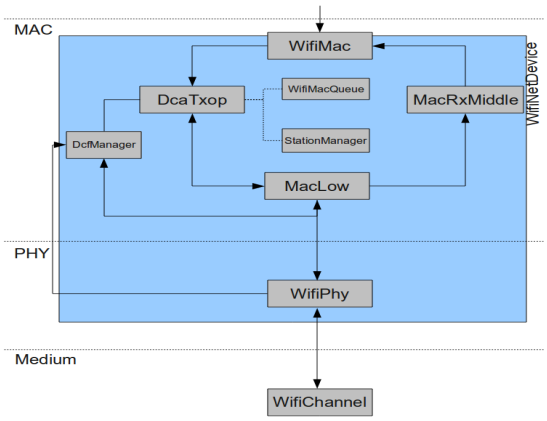

Fig. 3. Wifi module architecture

Each NetDevice is attached to a channel, over which it sends and receives packets. A Node may be connected to more than one Channel through multiple NetDevices. A node may use multiple protocol components, which handle packets received by the NetDevice. Each node can also handle a list of Applications.
The binding interfaces between the components of the node are designed to be similar to those in real systems. The interface between Application and Protocols is based on a class called Socket. This class is used by the application to send and receive traffic to the protocol stacks that are attached to a node.

- Packet : Packets are designed to have the same format as packets sent in real networks.
- Channel : A Channel can represent a wired network cable or a wireless transmission medium. Each Channel object may be connected to a list of NetDevice.

B. Architecture of IEEE 802.11 module

The implementation of IEEE 802.11 was ported to ns-3 from yans [7]. yans (Yet Another Network Simulator) was a prototype project by Mathieu Lacage and Tom Henderson. It implements a MAC and a physical layer that conform to the 802.11a specification. The focus of the MAC design is to model all the complexities in the IEEE 802.11 CSMA/CA mechanism which is known as DCF (Distributed Coordination Function). The 802.11 DCF is used to decide when to grant access to the medium. Its mechanism is presented briefly below.

1) 802.11 DCF access mechanism: The DCF access method is based on a CSMA/CA MAC protocol. Every node must sense the medium, to check the state of the channel (idle or busy). If a node has data to send, but it sees that the channel is busy, then it waits for the end of transmission.
At the end of transmission, it must wait for a DIFS (DCF Interframe Space). To prevent all nodes sensing the channel from beginning their transmission at the same time, each node chooses a random waiting time called Backoff before starting its data transmission. The range from which the Backoff value is chosen is called the Contention Window (CW). If the channel becomes busy during Backoff, Backoff counter decrementing is stopped until the channel becomes idle again.

DCF uses ACK packets to acknowledge the reception of unicast data transmission.

The DCF mechanism is also based on the use of the RTS/CTS (Request- To-Send and Clear-To-Send) scheme. If a node has a packet to send, it firstly transmits an RTS packet to request the channel. If the receiver is ready to receive data, it replies with a CTS packet. After the sender receives the CTS packet successfully, it transmits the actual data packet.

2) Wifi Module overview: Figure 3, shows an overview of 802.11 module architecture in the ns-3 simulator. If transmission is initiated by an application, the WifiNetDevice interface sends the packet to WifiMac class which handles high MAC level functions.

Each WifiNetDevice is aggregated to a StationManager which stores all transmission parameters (e.g. payload data rate, RTS/CTS, fragmentation, etc.) The main classes implementing the IEEE 802.11 DCF scheme are DcfManager and DcaTxop classes.

DcaTxop can handle only one packet. If other transmissions have been initiated, WifiMac pushes the packets to WifiMacQueue.

DcaTxop is used to handle the request access to the channel from DcfManager. When access is granted, DcaTxop pushes the packet to MacLow for transmission. MacLow initiates data transmission.

WifiPhy class is designed to support all physical layer related issues, such as channel sensing, SINR computation, interferences, etc. Its design is sufficiently generic so that it is able to support the implementation of different MAC designs on top.

YansWifiPhy is the implementation of the IEEE 802.11 physical layer. It models an additive Gaussian Noise Channel (AWGN) with cumulative noise handled by InterferenceHelper.

The abstract class WifiChannel is designed to model the radio signal transmission. YansWifiChannel is the only implementation modeling an IEEE 802.11 channel. WifiChannel

calculates the delay of received packets according to PropagationDelayModel. PropagationLossModel is used to calculate the reception power.

3) Implementation of DCF access mechanism: In this section, we describe in detail the process of computing DCF Backoff and channel access granting.

Each packet in WifiMacQueue is aggregated to a DcfState. DcfState is designed to keep track (such as Backoff, CW, ..) of the state needed for a single DCF function. DcfManager can handle a set of DcfStates. Its major function is to handle Backoff counting and schedule events, such as access granting, Backoff end, etc. For each packet in the queue, DcaTxop calls DcfManager::RequestAccess to request access to the medium. The channel can be in one of two possible states.
The channel is busy. In this case, a collision is notified, and the Backoff procedure is restarted.
Otherwise, the channel is idle. The function GetBackoffEndFor() is called to calculate the expected end of Backoff. If the Backoff has already expired, medium access is granted by calling DoNotifyAccessGranted(). Then, packet is pushed to MacLow for transmission.

If two local DcfState are expected to get access to the channel at the same time, the first registered DcfState gains access to the channel. DcfManager notifies other DcfStates of an internal collision via DoNotifyInternalCollision().

When medium access is granted, transmission parameters are requested from StationManager. They are signaled to the MacLow via MacLowTransmissionParameters. If fragmentation is enabled. It is handled by DcaTxop, which sends only the packet fragment to the MacLow. In the IEEE 802.11 standard, using acknowledgments is mandatory for unicast transmission. Received or lost packets are signaled to DcaTxop which handles retransmissions.

V. The EY-NPMA Module: Ey-Wifi for ns-3

In this section the modifications added to ns-3 to support EY-NPMA access rules are described. The purpose is to extend the existing code of the wifi MAC protocol to implement the EY-NPMA channel access mechanism, yielding Ey-Wifi.

Dcf Scheme and EY-NPMA access rules have common phases.

In Dcf, a node may transmit if and only if the medium is sensed to be idle during the Backoff interval. The main idea of the EY-NPMA implementation is to repeat this process twice. In the first phase, the node senses the medium. If the channel is idle, it sends a burst signal. At the end of the burst transmission, the node must sense the medium again and sends data packets if the channel is idle. The yield phase and data transmission in EY-NPMA can be considered as equivalent to Backoff countdown and grant access processing in DCF. The extra feature of EY-NPMA is the elimination phase.
As explained above, DcaTxop and DcfManager already handle many of the functionalities required.

We have opted to create a new “State“ which keeps track of the burst for a single elimination phase. EyState (for Elimination-Yield State) has been designed to handle the first two phases of EY-NPMA. Several modifications have been made mainly in DcaTxop , DcfManager and MacLow classes.

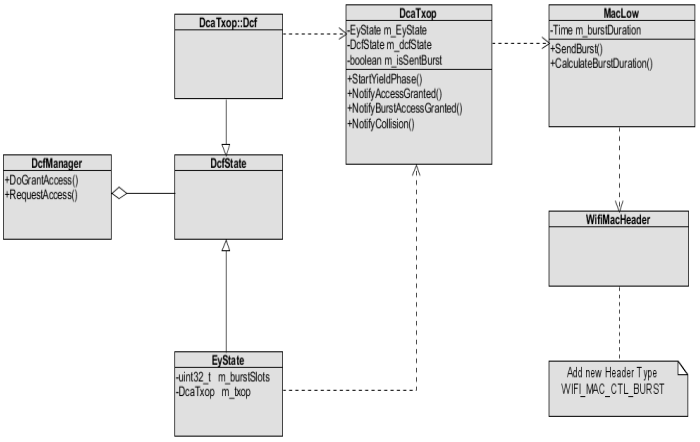

Fig. 4. UML diagram of EY-NPMA related classes

A. Operation sequence

In this section, we briefly describe the sequence of actions that take place during a simulation of the EY-NPMA protocol. In figure 4, the UML diagram shows an overview of EY-NPMA components. As seen in the figure, the module is composed of several C++ classes, each representing a component of the Ey-Wifi module and handling a particular functionality. This diagram only shows important attributes and functions.

EyState class inherits all methods and attributes from DcfState class. In addition to functionalities and tracks stored by DcfState, EyState handles the burst slots value.
For each packet in WifiQueue, two DcfStates are aggregated; An EyState to handle the burst transmission, and DcfState to handle Data transmission. When a node has a packet to send, DcaTxop requests access by calling DcfManager::RequestAccess(DcfState). If a burst has already been sent, RequestAccess takes a ”normal“ DcfState as parameter. Otherwise, RequestAccess(EyState) is called.
RequestAccess(EyState) checks if Backoff is fully elapsed (the Backoff here is equivalent to priority waiting time in priority phase) The same function GetBackoffEndFor() is called to calculate the expected end of Backoff. If access is requested, Backoff is fully elapsed and channel is idle, the DcaTxop::NotifyAccessGranted is called by DcfManager. This function has been modified, to support the transmission of the burst signal.

In NotifyAccessGranted(), the coordinator verifies whether the burst has not been already sent. If not, NotifyBurstAcessGranted() is called to inform MacLow that access is grant for burst transmission.

MacLow handles the creation and transmission of the burst signal. The burst signal is modeled by ns-3 packet object. The size of this burst packet depends on the random length

burst chosen by the node. CalculateBurstSize() is called to calculate burst size. Once the access is granted and burst size is calculated, SendBurst() initiates the transmission of the burst. At the end of transmission, StartYieldPhase() is scheduled. RequestAccess(DcfState) is called to request access for data. If the channel is idle, the node takes a random waiting time before transmitting its data. StartBackoffNow() is called to start a Backoff by initializing the Backoff counter to the number of yield slots specified. At the end of yield phase, DcaTxop checks again medium state. If channel is idle and Backoff is entirely elapsed, medium access is granted. Then, NotifyAccessGranted is called and data transmission is initiated.

On collision events, NotifyCollision and NotifyInternalCollision are invoked. Nodes, who quit the contention, must wait for the end of the current transmission. Request for access are rescheduled at the next predicted end of current transmission automatically.

For each phase, CWmin and CWmax (the range of Backoff) are set as mentioned in EY-NPMA standard.

- Priority phase : CWmin=0, CWmax=4
- Elimination phase : CWmin=0, CWmax=12
- Yield phase : CWmin=0, CWmax=9

B. Illustration

To illustrate the functioning of the MAC layer, a scenario containing 10 nodes was used, and is described in this section. The distance between the nodes is fixed to 5m. Nodes 0 and 1 periodically transmit UDP packets to node 9 at a contant rate. Unicast packets containing 1096 bytes are sent every 1s. If the packet is received, node 9 transmits an ACK to acknowledge the reception. This experiment is based on a physical layer operating at a bit rate of 6 Mbps. The simulation run executes for 10s and 20 packets are sent in total.

Figure 5 represents the channel access of the three main nodes (0, 1, and 9). The plots show the duration of transmission for each node (a value of 0 represents channel listening/receiving, a value of 1 represents transmission) depending on time. Node 0 and node 1 request access to the medium. Both of them send an elimination burst. Node 0 has the shortest burst. After transmitting its burst, it senses that the channel is busy. So, it quits the contention. The surviving node listens to the channel during a yield period. During this interval, the channel remains idle. Node 1 wins the contention and begins its data transmission. Node 9 sends an ACK packet in unicast to acknowledge the reception of data from node 1.

At the end of the transmission, node 0 tries to gain access to the channel again. It sends a burst signal, and then it waits during a yield interval. In this scenario, there are no other contending nodes, so it sends its data to node 9. When the data is received correctly by node 9, it transmits an ACK packet.

VI. Conclusion

This paper detailed the design and implementation of Ey-Wifi, an EY-NPMA module for ns-3 simulator. The proposed module implements the different phases of EY-NPMA channel access protocol. Based on existing Wifi implementation, this module implements key components of EY-NPMA channel access protocol. Benefiting from the features of ns-3, this module may enable tracing, emulation, and implementation of new scenarios.

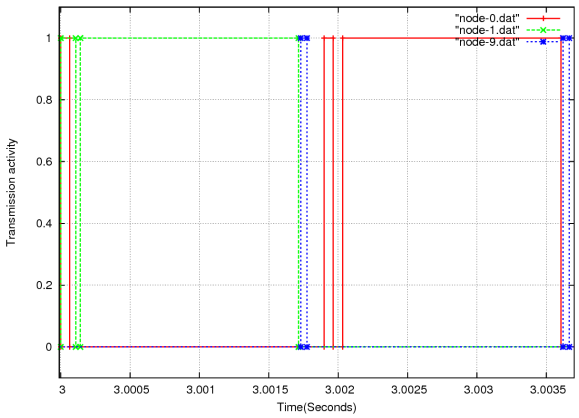

Fig. 5. Transmission activity of each node w.r.t time

There are few suitable simulation environments for EY-NPMA protocol. The purpose of this implementation is to facilitate evaluating and designing scenarios for EY-NPMA based wireless networks.

Acknowledgment

We would like to thank Y. Toor and A. Laouiti for contributing with their EY-NPMA work. We would also like to thank M. Lacage, D. Camara and W. Dabbous for their help and suggestions during the development of this work.

References

[1] EN 300 652 V1.2.1 (1998-07) *Broadband Radio Access Networks (BRAN) ; High PErformance Radio Local Area Network (HIPERLAN) Type 1* ; Functional specification, European Standard (Telecommunications series), ETSI. July 1998.

[2] IEEE 802.11 WG, "*IEEE Standard for Information Technology-Telecommunications and Information Exchange Between Systems-Local and Metropolitan Area Networks-Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications,*" IEEE Std. 802.11-2007 (Revision of IEEE Std. 802.11-1999), 2007.

[3] IEEE Std. 802.11 WG, "*Amendment to Standard for Information Technology. LAN/MAN Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Medium Access Control (MAC) Enhancements for Quality of Service (QoS),*" Nov. 2005.

[4] Philippe Jacquet, Pascale Minet, Paul Muhlethaler, and Nicolas Rivierre, "*Priority and Collision Detection with active Signaling: The Channel Access Mechanism of HIPERLAN,*", Wireless Personal Communications Vol 4, No 1, pp. 11-25, 1997.

[5] P. Muhlethaler, Y. Toor and A. Laouiti, "*Suitability of HIPERLAN's EY-NPMA for traffic jam scenarios in VANETs,*", ITST '10, Kyoto, Japan.

[6] A. Qayyum. *Analysis and evaluation of channel access schemes and routing protocols in wireless LANs*. PhD thesis, University of Paris-Sud, 2000.

[7] Lacage, M. and Henderson, T.R. *Yet another network simulator*, ACM Proceeding from the 2006 workshop on ns-2: the IP network simulator.

[8] Khosroshahy, M. and Turletti, T. and Obraczka, K. and others *Snapshot of MAC, PHY and Propagation Models for IEEE 802.11 in Open-Source Network Simulators*, 2007

[9] T. Bingmann. *Accuracy Enhancements of the 802.11 Model and EDCA QoS Extensions in ns-3*. PhD thesis, University of Karlsruhe, 2009.

[10] D. Dhoutaut. *Etude du standard IEEE 802.11 dans le cadre des reseaux ad hoc : de la simulation a l'experimentation*. PhD thesis, CITI Laboratory, INSA - Lyon, 2003

[11] The ns-3 network simulator, *http://www.nsnam.org/* , 2012