AALBORG UNIVERSITET

Department Of Electronic Systems

Simulation of Random Linear Network Coding in Ad-Hoc Networks

Final thesis Group 995 MOB

Beatriz García Segovia

31st May 2012
Supervisors: Stephan Rein
Frank Fitzek
Morten V. Pedersen

AALBORG UNIVERSITET

Department of Electronic Systems I-8
Fredrik Bajers Vej 7
9220 Aalborg Øst
http://www.es.aau.dk/

Title:
Simulation of Random Linear Network Coding in Ad-Hoc Networks

Theme:
WIRELESS MESH NETWORK.

Project Period:
10th Semester - Spring 2012

Project Group:
995

Member:

Beatriz García Segovia

Supervisors:
Stephan Rein
Frank Fitzek
Morten V. Pedersen

Number of copies printed and page:
5, 46

Finish 31st May 2012.

Synopsis:

The purpose of this project is to simulate random linear coding using NS3. To get this goal, it has been used NS3 as a library. In this way, it has been created a program which can be easily tested in NS3 without change the content of the simulator. Moreover, during this thesis it has been investigated the benefits of cooperative relaying in the Alice and BOB topology, as well as it has been investigated the benefits of using recoding in the relay.

Contents

List of Figures

Notations

X	is the new coded packet of size L (K/s symbols)
B	are the original packets, also called information vector
C	are the coding coefficients in Galois Field, also called encoding vector
j	implies the generation
i	has the maximum value of the number of packet to be combined
S	Number of transmitted packet from Alice
N	Number of received packet in Bob
N_R	Number of received packets on Bob from relay
Ni_R	Number of innovative from relay
P_i	probability of losses in the link A->B
P(i)	Probability of having a useful packet in the relay
ϵ_R	combination of ϵ_{AR} and ϵ_{RB}
ϵ_{AR}	is the probability of losses in link Alice-Relay
ϵ_{RB}	is the probability of losses in link Relay-Bob
ϵ_{AB}	is the probability of losses in link Alice-Bob
ρ	number of retransmissions
k	number of iterations in the sum operation

Chapter 1

Introduction

1.1 Introduction

Network coding has the potential to become a powerful tool for increasing performance in packet networks. The fundamental principle which it is based on is the potential to re-combine received packets in a node before forwarding it, instead of using traditional routing where the network nodes simply forward packets. In this way, this technique can help to improve the throughput by decreasing number of transmissions but obtaining the same number of packets in destination.

Network coding can be used in different topologies, for instance, the topology of Alice and Bob consist of three nodes where one is the receiver, another is the transceiver, and the third one is a relay. Figure 1.1 shows the difference in this topology between using traditional routing, above in the figure, and using network coding, below in the figure. Above can be observed that to sent a packet from Alice and Bob using a relay, has to be used 2 transmissions, once from Alice to relay and another one from relay to Bob. The same case can be observed when communication flows between Bob and Alice. Therefore, 4 transmissions are needed in order to maintain the communication using traditional routing. However, below in Figure 1.1, it can be observed that to maintain the communication using network coding in relay, it can be saved one transmission. Therefore, using network coding in this topology can give a theoretical throughput gain of 25%.

In Figure 1.1 is used simple XOR coding, which is commonly known as inter-flow network coding. Inter-flow network coding operates combining packets from different sources. However, in our scenario all the combined packets are from the same source, therefore in this thesis is going to be used intra-flow network coding.

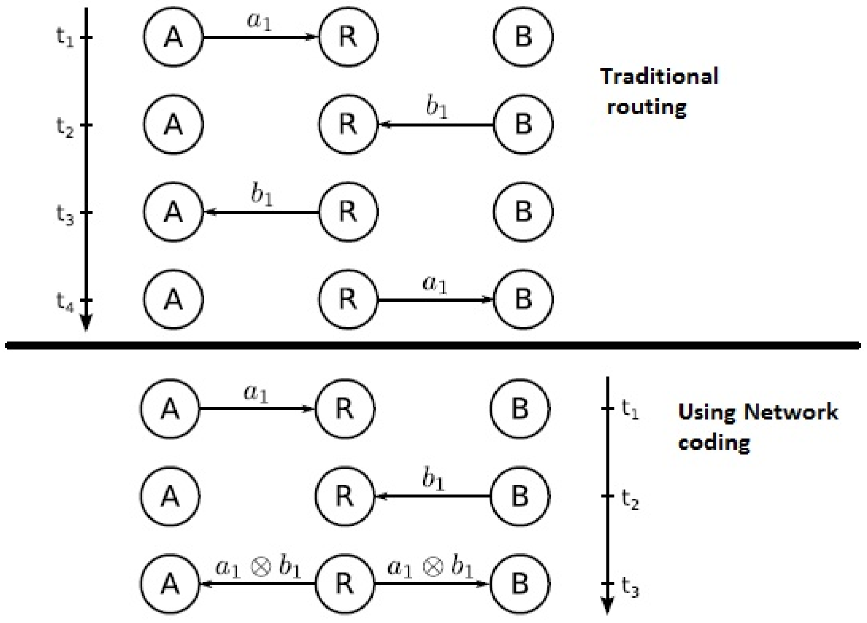

Figure 1.1: Transmission using traditional routing versus using Network coding [IFN11]

The coding presented in Figure 1.1 relies on simple XOR coding, between two separate flows which has been successfully implemented in CATWOMAN [IFN11]. In this report is going to be investigate the use of network coding in a different scenario, which it will be called cooperative relaying where coding is performed within a single flow. The basic idea in cooperative relaying is shown in Figure 1.2, where the relay will improve the communication between Alice and Bob when the main link between them is bad. Using simulations and analytical analysis it is going to be investigated the usefulness of cooperative relaying in the used topology.
Like CATWOMAN [IFN11], the implemented protocols of this thesis are going to be developed in MAC Layer. This fact is due to MAC layer provides addressing and channel access control mechanisms that make it possible the communication between nodes. Moreover, implementing this at the MAC layer also has the advantage that existing system typically based on TCP/IP or UDP/IP do not have to be changed.
This thesis also investigates how this topology can be simulated in the NS3 network simulator. NS3 is a discrete-event network simulator for Internet systems. NS3 is free software, licensed under the GNU GPLv2 license.

1.2 Motivation

Nowadays wireless devices are more and more popular. Furthermore, the increased amount of traffic generated by new services means that the networks are becoming

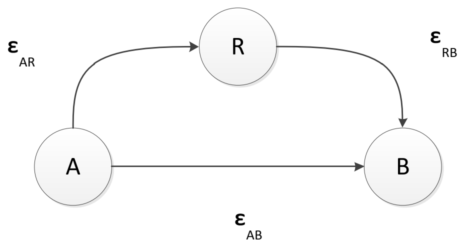

Figure 1.2: Alice and Bob topology for the cooperative relay

increasingly congested. Moreover, wireless devices are supplied by batteries, therefore caring about energy consumption is something to take into account. For these reasons it is necessary to continuously improve protocols and investigating new paths to obtain an effective use of the available resources, saving the maximum amount of energy and developing fast and reliable systems.

The simple relay model suggested could be applicable in a wise range of wireless networks, for instance mesh networks but also in standard 802.11 infrastructure model networks. Therefore, this topology service is a good starting point to investigate the intra-flow network coding.

We will attempt to implement the proposed scenario in the NS3 network simulator. NS3 was chosen in order to obtained robustness and throughput improvement, since it provides a interface simulation where it is possible to filter all kind of errors and find possible solutions which fit with the goal of the code. Besides, as free software, this thesis can contribute to the community and set the basis of network coding in NS3 for future developers.

1.3 Problem definition

The main goal of this thesis is to investigate if cooperative relaying can help in the communication between two nodes, where cooperative relaying will be implemented using intra-flow network coding and studying the benefits of using recoded packets in the relay.
Also it is desired to evaluate the use of the NS3 network simulator as a platform for implementing and simulating network coding based systems and protocols.
We will evaluate this through the implementation of simple protocols which will be tested the use of cooperative relaying, and intra-flow network coding.

1.4 Project Overview

This report has been split in 5 chapters and one for the appendix. During this chapter, a small introduction about the topic was made. In addition, the problem definition and motivation have been presented. In the second chapter there are explained some basics concepts used during this thesis, as random linear network coding, and an introduction to NS3 simulator. In the third chapter, itis going to be explained an overview of the implementation. In the fourth chapter, are going to be explained the protocols what have been studied and simulated, as well as the comparison between the analytical results and the simulations.

In the last chapter will be presented the conclusions and future works and in the appendix is going to be explained how to use NS3 as a library to build our system out of the simulator.

1.5 Acknowledgements

I do not have enough words to thank the effort and the patience than Morten V. Pedersen and Peyman Pahlavani have had with me during the course of this thesis. Their continued support and wise advice have made the difference in this report. Thank you to make me feel as part of a team, despite my group team was just me.

1.6 Notes and references

For the bibliography it is used cites, which will be included at the end of the paragraph, in the beginning of the section or in the caption. The bibliography will be cited using the first 3 letters of the author surname and year of the publications if this is included in the bibliography.

To get this report clear and easy to follow, it is established the notations that will be used, which list can be read in the beginning of this document.

Chapter 2

Background

2.1 Random Linear Network Coding

[NCO09]

There are two main approaches for implementing intra-flow network coding: Deterministic Network coding and Random Linear Network Coding (RLNC). Deterministic network coding is suitable for static topologies, therefore it does not fit for dynamic wireless channel as the channels used in our scenario. For this reason, this thesis is focused in RLNC. In this section, it will be explained which are the main issues of this concept and how it is used in the project environment.

2.2 Definition

[NCO09]

Network Coding is an in-network data processing technique that exploits the characteristics of the wireless medium in order to improve the throughput of the network.

2.2.1 Terminology related with Random Linear Network Coding

[NCO09] Before to focus in the main processes carried out in Network Coding, it would be explained some important concept, which will be used along this thesis:

- Link: The channel where the packets are forwarded between two nodes.

- Generation: In order to do feasible network coding, it is needed to create groups of packets called generations, where only packets of the same generation can be linearly combined. Each packet is assumed to consist of symbols over a finite field.
- Galois Field: is a field that contains a finite number of elements.
- Coefficients: Choosing the coding coefficients could be a tough task. In order to solve this problem, during this thesis is selected random coefficients over Galois Field, using a uniform distribution.

RLNC consists in three main processes: encoding, recoding and decoding.

2.2.2 Encoding

[NCO09]

This is one of the main processes carried out in Network coding. Encoding needs to be performed only at source nodes of the network. It is based on the formula :

$$X_j = \sum_{i=1}^{N} C_{ij} \cdot B_i$$

Where,
X is the new coded packet of size L (K/s symbols),
B are the original packets, also called information vector
C are the coding coefficients in Galois Field, also called encoding vector
j implies the generation,
i has the maximum value of the number of packet to be combined.

Moreover, this expression can be calculate with a matrix multiplication, where the coding coefficients(C) and the original packets(B) are matrices. The matrix expresion is:

$$X = C \times B$$

2.2.3 Recoding

Recoding is performed at relay nodes of the network and is similar to the encoding process. However, it is more complicated in comparison with encoding, because the packets are already coded and the new coefficients has to be a linear combination of the old coefficients.

2.2.4 Decoding

[NCO09] Decoding needs to be performed only at received nodes of the network. When the destination node receives a set of encoded packets, it is required to obtain the original ones. For that reason, it is needed to solve the next equation:

$$B = C^{-1} \times X$$

This is a linear system with M equations and N unknowns. It is needed $M \geq N$ to achieve decoding data. However, that is not enough, the combinations might be linearly dependent also.

A packet is called innovative if it carries new information. The previous equation is a matricidal equation which in practice is solve as follow:

- The encoding and information vectors are received with the packets and stores in a matrix, which is called decoding matrix.
- Each encoded packet received is stores in the last row of the matrix.
- Each non-innovative packet is reduced to a row of 0s by Gaussian elimination and is ignored.

In this way, the matrix can be transformed to a reduced row echelon form, where all entries in a column below a leading entry are zeros.

2.2.5 How works RLNC

[NCO09] In this thesis RLNC will be applied RLNC in the following way, as is shown in Figure 2.1:

- Alice generates several packets.
- Buffering them until a generation is collected in the queue.
- Once the generation is collected all these packets are encoded in one packet with the same size, using linear coefficients which belongs to the Galois Field.
- This packet is forwarded.
- The relay node receives the packets, recoding them if it is needed it and forward them, maintaining, in this way, the flown information in the network. If there are too much intermediated nodes the probability of receive the packet increase.

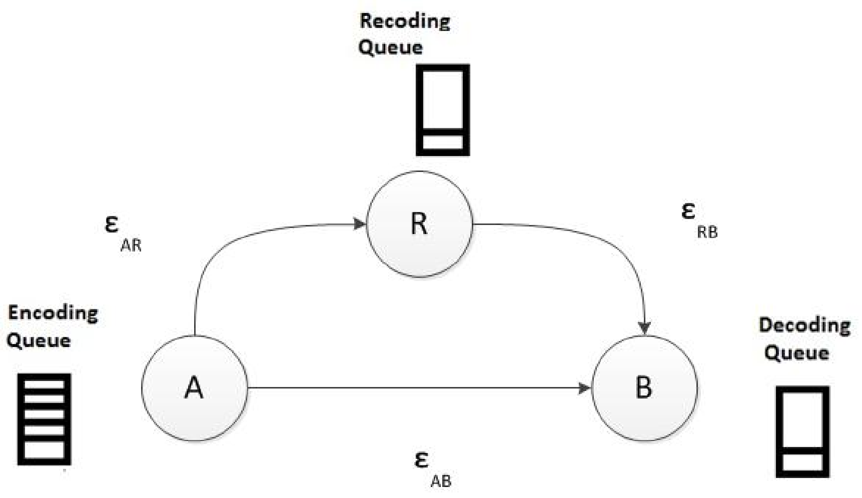

Figure 2.1: RLNC in the project scenario

- When Bob receives some packets, it discards repeated packets or corrupted, and starts decoding.

A significant difference between a coded packet by linear combination and a concatenation of packet is that despite both carries information about originals packets, the coded packets just by itself does not allow to recover any part of the original ones.

2.3 NS3 Simulator

[8BW10]

This section describes how to use NS-3 Network Simulator and how to implement RLNC into NS3 simulation. With the code that it will be generated, it will be simulate a mesh network with the same behavior and parameters as it can be found in the real world. Therefore, it is pretended to obtain a simulation as similar to real network using random linear network coding as possible.

The NS-3 Network Simulator is specially focused on Internet-based systems. In this simulator, codes and library components are written in C++, although is also possible to write them in Python. In the next sections are going to be explained some codes which have been generated in order to simulate the desired environment.

The intention of use object-oriented languages and use a GNU license is to allow to the user to create or modify models, which enrich the simulator itself.

NS3 has a modular architecture and is based on the NS3 supported libraries and other ones that can be added by users. This libraries can belong to :

- Core library: For generic aspects, such as callbacks, debugging objects, etc.
- Simulator library: Schedules, events,etc.
- Common library: For independent objects as packets.
- Internet: Models and protocols related to internet such as TCP/UDP.
- Node library: Consist of abstract classes.

2.3.1 OSI Model

The Open System Interconnection (OSI) model is an effort at the International community in order to define a networking framework for implementing protocols in abstract layers. Achieving a system where is possible to work in a layer without taking into account what happens above or below, is a goal of this model.

In our simulation, the information is generated and forwarded into the different layers, but this thesis is going to be focus in the MAC layer, because is where the code is set up. Figure 2.2 shows how the information flows in our simulation and which kind of protocols are going to be found to work with them, in the MAC layer.

2.3.2 NS3 802.11 implementation

[8BW10]

The starting point to start programing is going to be the WIFI model developed in NS3. This model is called WifiNetDevice, and next it is going to be shown its behavior.

Firstly, it is needed to talk about the simulation objects:

- Class Node can be considered as a base class in NS3. However, users are allowed to create their own Node subclasses.

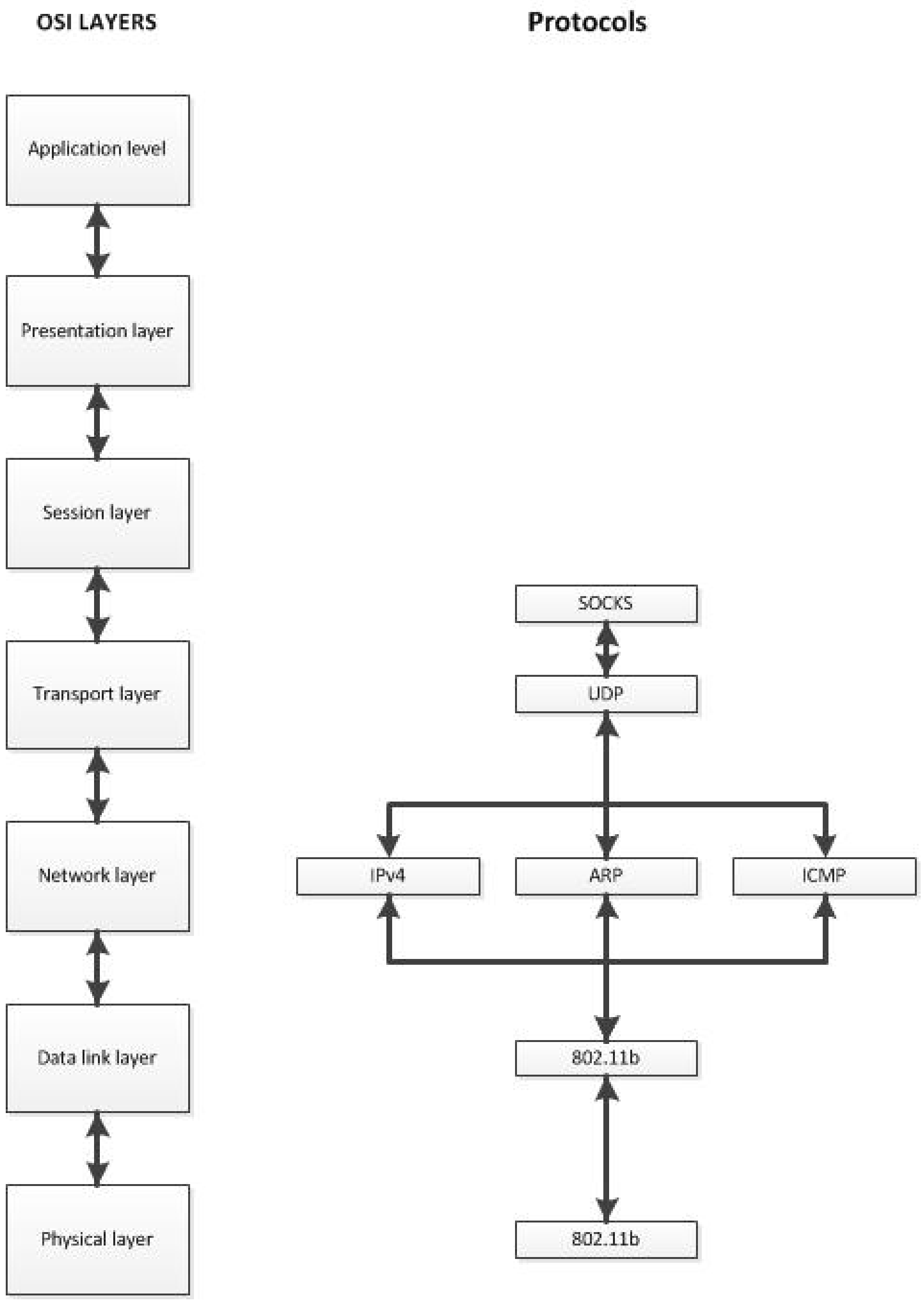

Figure 2.2: OSI model and protocols

- Class NetDevice represents a physical interface on a node (such as an Ethernet interface).
- Class Channel, which is closely coupled to the attached NetDevices, implements a logical path over which the information flows.
- NS3 Packet objects contain a buffer of bytes where protocol headers and trailers are serialized. The content of this buffer has to be the same as the content of a real packet on a real network in the desired scenario. The design of this object has been done in order to:
  - Avoid changing the core, with different kind of packets.
  - Maximize the integration with real-world code and systems.
  - Make it easy the fragmentation, defragmentation and concatenation processes.
  - Make a efficiently use of memory.
- Applications generate traffic which it is needed in order to make simulations of the network. Applications work down the protocol stack and maintain the communication between nodes via sockets.

Now that these objects have been presented, it is possible to talk about the model WifiNetDevice based on the IEEE 802.11 standard.

2.3.3 WifiNetDevice

[311b]

Configuration of WifiNetDevice is complex. For this reason, NS3 provides some helper classes to perform common operations in a simple matter, and leverage the NS3 attribute system to allow users to control the parametrization of the underlying models. [311b]

In order to use a WifiNetDevice to a node, it must create an instance of a WifiNetDevice and a number of constituent objects.

The generate code during this project is based in this structure, being modify only the functions send and forwardUp and being create some others functions which calls are made to/from this two functions. Above in Figure 2.3, it can be observe the position of both functions in the WifiNetDevice architecture.

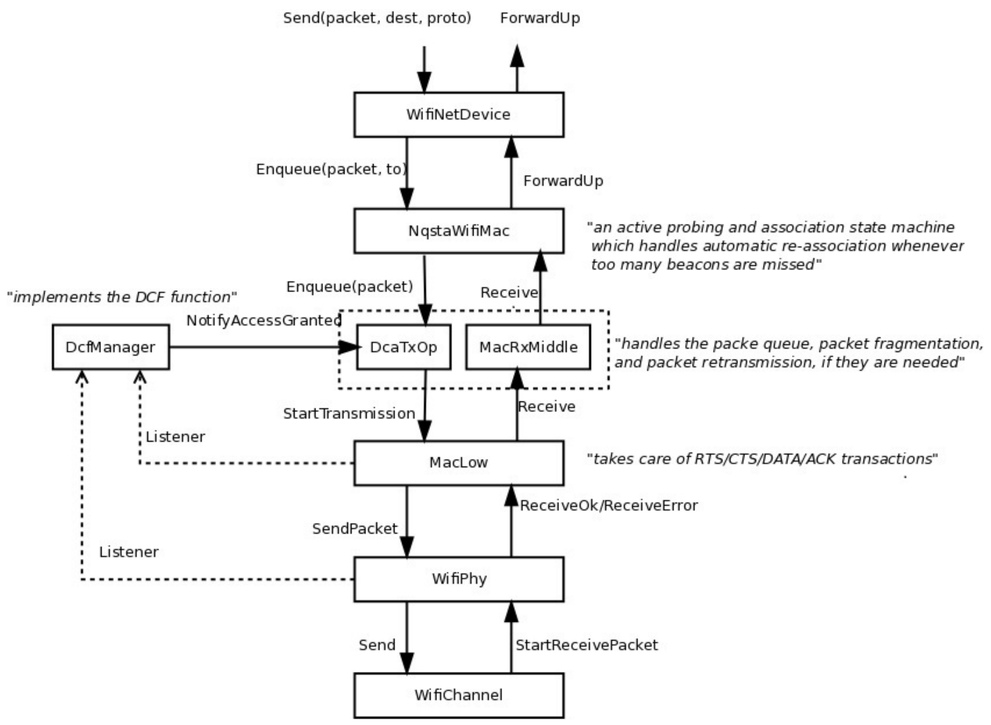
Send(packet, dest, proto)
ForwardUp
WifiNetDevice
Enqueue(packet, to)
ForwardUp
NqstaWifiMac
"an active probing and association state machine which handles automatic re-association whenever too many beacons are missed"
"implements the DCF function"
Enqueue(packet)
Receive
NotifyAccessGranted
DcfManager
DcaTxOp
MacRxMiddle
"handles the packe queue, packet fragmentation, and packet retransmission, if they are needed"
StartTransmission
Receive
Listener
MacLow
"takes care of RTS/CTS/DATA/ACK transactions"
ReceiveOk/ReceiveError
SendPacket
Listener
WifiPhy
Send
StartReceivePacket
WifiChannel

Figure 2.3: WifiNetDevice architecture [311b]

2.3.4 Propagation models

One of the most important parameters in order to do an accurate simulation is the propagation loss. Due to interferences and obstacles and the phenomena which they provoke, the signal strength decreases in different ways, and this fact must be taken into account in simulations.

Due to the limitations in time and resources, it has been chosen a simple propagation models where the only factor which matters is the distance, however, in future develops, it has to be chosen or maybe even developed an appropriate propagation loss model in order to achieved more realistic results. Figure 2.4 shows the models currently available models in NS3.

2.3.5 How NS3 produces usable programs

Once the NS3 code is placed in a local system, it is needed to compile this code, in order to create the desired programs. For this goal, NS3 uses Waf which is a Python-based framework for configuring, compiling and installing applications.

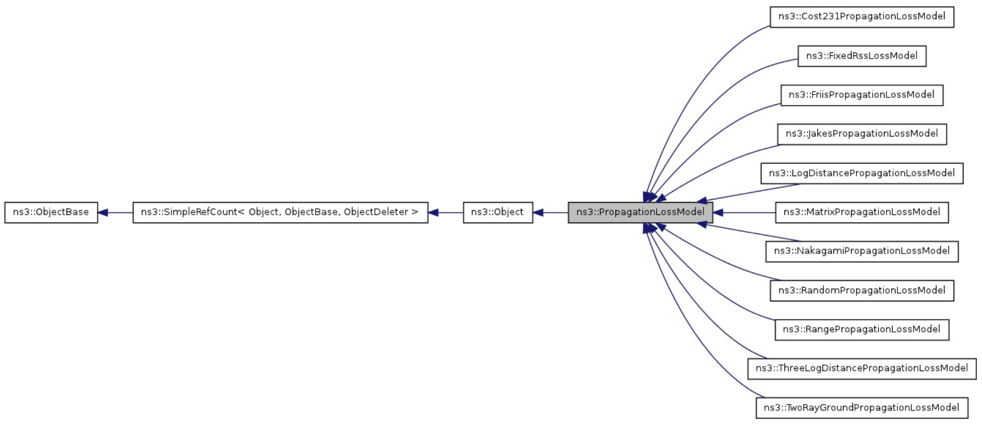
ns3::ObjectBase
ns3::SimpleRefCount< Object, ObjectBase, ObjectDeleter >
ns3::Object
ns3::PropagationLossModel
ns3::Cost231PropagationLossModel
ns3::FixedRssLossModel
ns3::FriisPropagationLossModel
ns3::JakesPropagationLossModel
ns3::LogDistancePropagationLossModel
ns3::MatrixPropagationLossModel
ns3::NakagamiPropagationLossModel
ns3::RandomPropagationLossModel
ns3::RangePropagationLossModel
ns3::ThreeLogDistancePropagationLossModel
ns3::TwoRayGroundPropagationLossModel

Figure 2.4: Propagation loss models [311a]

In order to build a program in a folder, it is required to have the Waf code and a wscript in the same location. This folders are included in NS3 installation folder, however, if it is required to build and configure a program outside NS3 it is recommended to download from [waf11].

Wscripts are python scripts containing functions and variables that may be used by Waf. The commands of description files are simple functions. As it was said previously, the wscript should be in the same folder that the Waf code and c++ code. However, it is possible to create wscript in subfolders, which will be called by the top-level wscript. For this reason, in the NS3 folder can be found several wscripts in different folders, but all of them are call from Waf placed in the main NS3 folder, when it is required.

Chapter 3

Implementation

3.1 Introduction

As it has been mentioned in previous sections, this thesis is going to work in MAC layer. Moreover, NS3 is going to be used as simulator and it is needed to set up the scenario where we will work. In order to get this goal, some c++ codes and classes have been developed and used as top of the WifiNetDevice class which was introduced in the previous chapter.

The name of the new class is PepWifiNetDevice and it is used to hold on the packets, doing some operations with the information (coding, recoding, decoding) and forward up the information to the upper layers.

3.2 Main processes

The main processes are coding in source, recoding in relay and decoding in the sink. In Figures 3.1, 3.2 and 3.3 can be observed how it is treated the information when it reaches one of these nodes.

3.2.1 Source

In the source, it is going to be done the coding. In case that the packet which is going to be sent is an ARP packet, it will be sent without encoding.
In order to encode the packets, they are going to be stored in a queue until it reaches

the generation number of packets in there. Once this is accomplished, the payload of the packet is extracted, encoded with the Galois coefficients and create a new packet with the result of encoding, which will be sent to destination. The basic flow diagram in the source is shown in Figure 3.1.

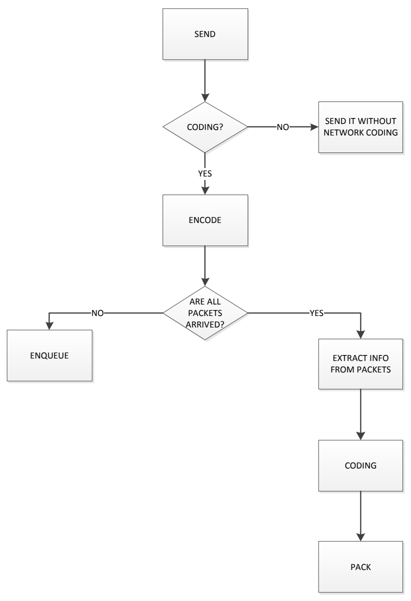

Figure 3.1: Structure of processes in Source

3.2.2 Relay

In the relay, it can occur two facts: the packet is just forwarded or the packet is recoding and then forwarded.
Moreover, the relay has a buffer to storage all the packets that reaches it, and this packets will be used in the recoding over and over again. The basic flow diagram in the relay is shown in Figure 3.2.

3.2.3 Sink

In the sink, it is going to be decoded the packets and also it is going to be send the ACK packets to the source when a complete generation is decoded. The basic flow diagram in the sink is shown in Figure 3.3.

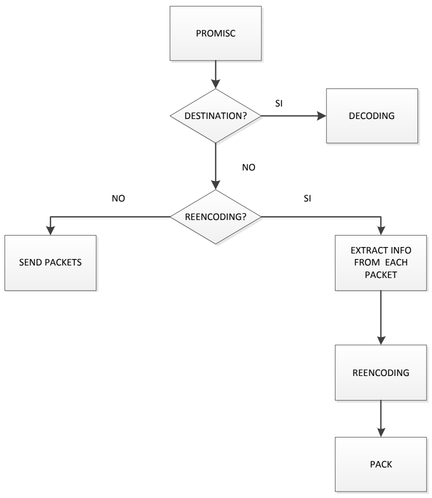

Figure 3.2: Structure of processes in Relay

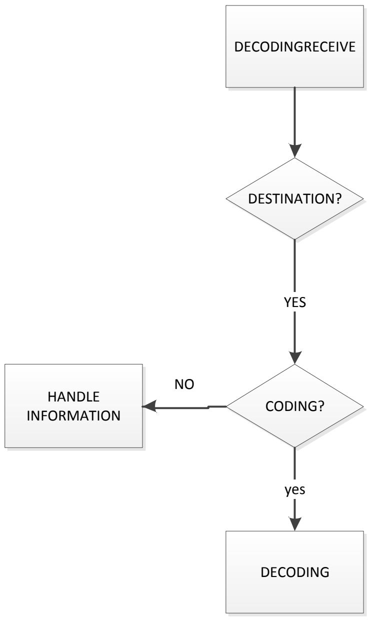

Figure 3.3: Structure of processes in Destination

Chapter 4

Protocols Description

Wireless mesh networks is a suitable platform to use network coding. In this kind of network, each node helps the communication increasing the information flow in the network, due to each node receives and forwards packets. Since the wireless channel is spread, the paths built by the routing protocols are artificial in the sense that more than the source and the destination will overhear a specific packet, these nodes are potential relays. However, the information flows in different routes, and it can occurs that one node receives the same packet from two different paths. For this reason, it want to be studied when the relay activity is worth it and when it can help to the communication between two nodes and when it is just a resources wasted.

In addition, this fact can be apply to other kind of networks, where wireless communication is involved and somehow it can help the communication between two devices.

4.1 Scenario

In this context, it is set up the first scenario where testing will be done. The situation is like this: two nodes, Alice and Bob, which are placed far away each other, want to maintain a communication. Some packets are dropped because interferences, and as it is used random linear network coding, some generation can not be decoded. That means more losses in the network. However, there is a third node which has a good quality link with both nodes. In this situation, the third node, called relay, helps the communication forwarding the packets which it receives and belongs to one of the other nodes. An example of this kind of situation can be observe in the picture 1.2

It has to be mentioned than all the nodes in the network are setting in promiscuous mode. Promiscuous mode means that the communication between two nodes is unicast, but the surrounded nodes can overhear the information that is flown between these nodes.

During the simulation, it is going to be used random linear coding, such as a packet to be send it has to be a combination of several packets. Moreover, in the relay it is used recoding for increase the received packet probability in the destination. However, this fact must be used with criteria because this process can make decoding process more complex or even impossible.

4.1.1 Advantages and disadvantages of recoding in relay

When a packet is received by the relay, it can just forward the packet, or recode the packets and then forward it. In this basic protocol it is going to be tested which are the benefits of these actions, in to order to achieve more accurate protocols in the future.

The main advantage of using recoding is that it can be increased the number of innovative packets in the destination. However, this fact has to be balance with the fact of this process needs energy and time, so it introduces a delay in the arrival packets.

4.2 Protocols in the network

In this thesis has been studied three cases:

1. Simplest case: two nodes, source where coding is taking place and sink where is made decoding.
2. Cooperative relaying case: three nodes, source where coding is taking place, sink where is made decoding and where relay just forwards the packets.
3. Cooperative relaying + recode case: three nodes, source where coding is taking place, sink where is made decoding and where relay recodes and forwards the packets.

4.2.1 Protocol 1: Simplest case

This simplest case has been studied in order to compare the throughput obtained here with the results obtained in cooperative relaying case protocols. The scenario of this protocol is explained in Figure 4.1.

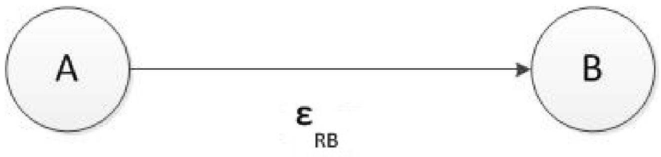

Figure 4.1: Communication between Alice and Bob

4.2.1.1 Analytical calculations

In this context, the number of packets S that have to be transmitted by Alice, to achieve N packets in Bob, can be calculated as:

$$S = \frac{N}{1 - \epsilon_{AB}}$$

4.2.2 Protocol 2: Cooperative relaying case

Coming back to our scenario, it can be found that there are 3 nodes:

- Alice
- Relay
- Bob

In this case, when a packet is received in the relay, this packet is just forwarded. Figure 4.2 shows the error probability (ϵ) of each link.

4.2.2.1 Analytical calculations

In this context, the number of received packets in Bob from relay can be calculated as:

$$N_R = S \cdot (1 - \epsilon_{AR}) \cdot (1 - \epsilon_{RB})$$

The probability of an innovative packets is arrived from relay to destination, is an important issue to study in this thesis, because this value can help to make decisions in the behavior of the relay. Taking into account this fact and the previous equation, it can be found that the number of innovative packets from relay can be calculated as:

$$Ni_R = S \cdot \epsilon_{AB} \cdot (1 - \epsilon_{AR}) \cdot (1 - \epsilon_{RB})$$

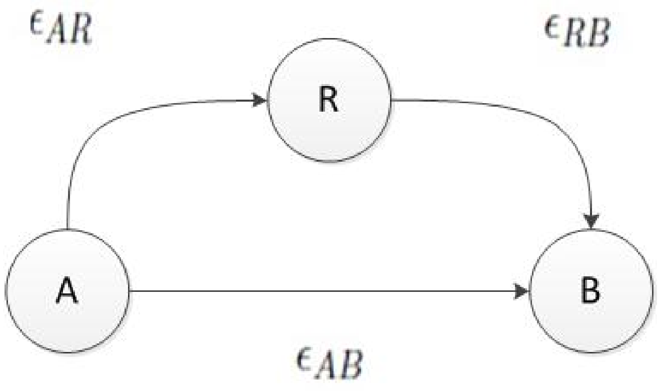

Figure 4.2: Scenario Alice, Bob and Relay

where it is considered:

$$\epsilon_R = -\epsilon_{AR} - \epsilon_{RB} + \epsilon_{AR} \cdot \epsilon_{RB}$$

Therefore,

$$Ni_R = S \cdot \epsilon_{AB} \cdot (1 - \epsilon_R)$$

And the number of packets that have to be transmitted from Alice to achieve N packets in Bob are calculated as:

$$N - Ni_R = S \cdot (1 - \epsilon_{AB})$$

$$Ni_R = S \cdot \epsilon_{AB} \cdot (1 - \epsilon_R)$$

$$S = \frac{N}{(1 - \epsilon_{AB}) + \epsilon_{AB} \cdot (1 - \epsilon_R)}$$

4.2.3 Protocol 3: Cooperative relaying + recode case

Figure 4.2 shows the scenario for this case, but in this occasion it is added the recoding processes in the relay.

4.2.3.1 Analytical calculations

In this context, the number of receiving packets in Bob from relay can be calculated as:

$$N_R = S \cdot (1 - \epsilon_{AR}) \cdot (1 - \epsilon_{RB})$$

In this scenario the calculation of innovative packet is more complicate than previous case. For this reason, it has been developed a formula, that if it is not completely

exact it can be considered really accurate. When in the relay is applied recoding, the probability to obtain an innovative packet depends on the link quality between the nodes, but also it depends on the forwarded history. If one packet reaches the relay, is storage in a buffer where it will be recoded together with the rest of the packets store in this buffer, creating a new packet that it will be forward to Bob. For these reason, when the number of transmission packet increases, it also increases the probability of obtaining an innovative packet which it has been lost in previous transmission but that is contained in the buffer.
In order to get the probability that a packet contained in the relay is useful it has been used some iterations:

1. First packet is received in the relay. There is nothing useful in the history because it is just the first packet.
2. Second packet is received in the relay. Here the history is useful when it was received a packet, but it was lost or not forwarded.
3. Third packet is received in the relay. Two fact can occur to get at least one useful packet in the history; either it was received the first packet and it was not forwarded or lost in the last two transmissions or the other case is that it was received the second packet but it was lost or not forwarded in the last transmission.
4. Fourth packet is received in the relay. Now, there are 3 possibilities:
    - the first packet was received but it was lost or not forwarded during the last three transmissions.
    - the second packet was received but it was lost or not forwarded during the last two transmissions.
    - the third packet was received but it was lost or not forwarded during the last transmissions.

In this way, it can be achieved the following formula to calculate the probability of having a useful packet in the relay:

$$P(i) = \sum_{j=0}^{j<i} (1 - \epsilon_{AR}) \cdot \epsilon_{RB}^{i-j}$$

This formula has been used in order to get the number of innovative packets on Bob as follow:

$$Ni_R(i) = \epsilon_{AB} \cdot (1-\epsilon_{AR}) \cdot (1-\epsilon_{RB}) + P(i) \cdot (1-\epsilon_{AB}) \cdot (1-\epsilon_{AR}) \cdot (1-\epsilon_{RB}) \cdot (1-(\epsilon_{AB} \cdot (1-\epsilon_{AR}) \cdot (1-\epsilon_{RB})))$$

This formula has been used in order to calculate the probability of get a innovative packet in Bob from both paths as follow:

$$P_i = (1 - \epsilon_{AB}) + Ni_R(i)$$

where i means the number of packets.
Now, in order to obtained the number of packet that has to be transmitted by Alice to obtain N packets on Bob, it is used the next approximation:

$$\sum_{i=0}^{S} P_i < N$$

Where S is the number of packets that have to be transmitted by Alice.

4.2.4 Analytical results

Figure 4.3 shows the theoretical number of packets that have to be transmitted by Alice in order to achieve a fixed number of packet in Bob, depending on the error probability on the main link, or said with other words, depending on the link quality between Alice and Bob, in the case shows in Figure 4.3 with the blue *, and also depending on the error probability on the Alice-Relay-Bob link, for the cases show in Figure 4.3 with red and green *. It has to be mentioned that green * has been obtained using recode in the relay, saving an average of 9% of trasnmissions, in this way.

These plots has been obtained using a 74% of error probability in link Alice-Bob and a 57% in the link Alice-Relay-Bob. The reason of that is that it has been used the same values, that it have been used during the simulations. After some time developing the scenario, and make it works, it have been observed that playing with this values in the simulation is a tough task and it has not been possible to make the simulations with different values in time to handle this report.

4.2.5 Simulation

Figure 4.3 shows the NS3 simulation of the relation between the number of transmitted packets from Alice and the number of received packets in Bob depending on the link quality between them, shows in Figure 4.3with blue line, between them and the relay, shows in Figure 4.3 with red line, and also using recoding in the relay, shows in Figure 4.3 with green line, after 9 simulations.

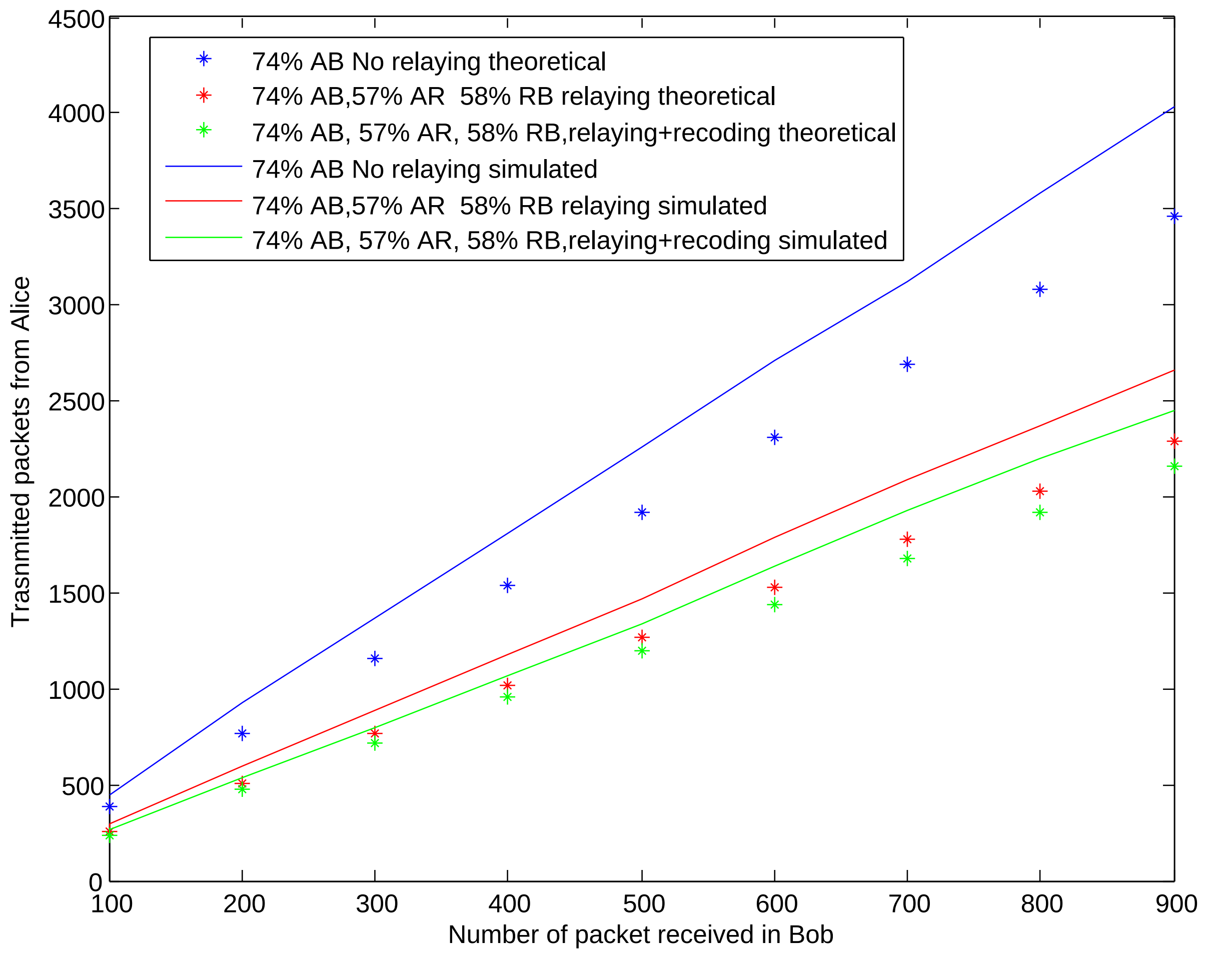

Figure 4.3: Number of transmitted packet from Alice simulated by NS3 and by analytical calculation

4.2.5.1 Discussion

NS3 is an environment which it is pretended to simulate a real network. Therefore, NS3 generates a simulated traffic as similar to real world as is possible. In this context, it can be explained the differences between the analytical results and the simulated results.

Figure 4.3 shows how if the number of received packets increases in Bob, also is increasing the number of transmission from Alice. Also Figure 4.3 shows that when the amount of packets received in Bob is low, the analytical result are close to the simulated results. This fact is due to NS3 tries to make a real network where if the number of packets flows in the network is high, there will be more problems with the packets like collisions and therefore, Alice will do more transmission. Therefore, there are differences between the analytical results and simulated results due to in simulations are included more factors which can affect in the packet behavior.

In respect to simulations, Figure fig:sim shows that the number of transmission from Alice in red line, decreases to achieve the same amount of packet in Bob, in comparison with blue line, that was shown in the case where the cooperative relaying is not used. Also, it can be compared in Figure 4.3 the green and the red lines, and check how using recoding on the relay, it can be improve the throughput gain in a 5.7% by saving transmissions.

Also it was achieved than when there are more packets in the network, the relay will be more efficient and it will forward more innovative packets due to its history.

4.2.6 Improving protocol

Once this environment is developed and tested, the protocol can be improve in order to get a more efficient green system and save energy. In this case, the relay has to make a choice between forward packets or ignore them, depending on the link quality where the communication is pretended to be maintained. For this reason, some limits have to be set in order to discern where to place the bounded, to achieved a good QoS.

Moreover, the relay also can decided if it is good for the network recoding the packets or just forward them, taking into account also the quality required in the network and the amount of information flow in there.

Once, all of these goals are accomplished, the scenario can be expanded and then can be simulated a network more realistic, using several sources and destinations.

4.3 Contribution of this Project

During this thesis has been simulated some cases of Random Linear Network Coding in a multipath network. When a suitable result will be achieved, the generate code use together NS3 libraries to make some simulations, which it is possible since NS3 has a GNU license. Therefore, this thesis will contribute to the community making possible the simulation of network coding using NS3 simulator libraries.

In fact, this contribution will make possible that other people use this code like a starting point for future investigation of network coding in multiple situations and topologies. Moreover, all these simulations and codes have been documented making easy the understanding of how works NS3, how to create suitable codes in this environment and how Random Linear Network Coding works in a multipath network.

4.4 Assumptions

Since there are limitations in time and resources, and in order to make the problem in this project more feasible and accessible, the following delimitations and assumptions are made.

- Same packet size during the simulation. This decision was make in order to simplify the code and be more focus on its behavior. If the packet size is different in each packet, it is needed somehow a dynamic stored, that means, it is needed a buffer with the packet size of each stored packet. In order to solve this situation, some paths can be follow:

  - Link together all the packets in an array with fixed size.
  - Create a matrix with a fixed number of position and maximum position size and add a field where the exactly size of each packet which is placed in this position, is added.

  In both cases padding with zeros will be needed in the remaining positions.

- The number of nodes during the simulations is two or three (depending on the protocol), considering fixed the destination and source nodes.
- Propagation losses only depends on distance.

Chapter 5

Future work and conclusions

5.1 Conclusions

To sump up all results obtained in this report, firstly it is going to be explained briefly what it has done during this project:

1. It has been implemented some protocols in order to incorporate network coding to NS3. This fact is remarkable since until now, no one has been done that before as far as our knowledge reaches. In order to get this goal, it have been used NS3 libraries to simulate our scenario and some results have been obtained.
2. It has been studied the benefits of using cooperative relaying in a network.
3. Also it has been studied the benefits of using recoding in the relay.

Therefore, it has been studied some issues and some conclusions can be achieved, these are:

- It has been set up the basis for future developers of network coding on NS3. This is a good starting point, due to the fact that NS3 is a powerful tool where it can be simulated different scenarios.
- It has been discovered than the utilization of the relay is worth it as far as the main link quality in the communication is worse than the link between the nodes and the relay. This fact has been studied theoretically, and also it has been done one simulation, however due to the complexity of the simulations focusing in link quality parameters and the lack of time, it was not possible to make more simulations in order to studied the effect of the link quality in the utilization of the relay. However, these tests can be achieved by future developers easily, now than the environment is set up.

- It has been found that using recoding in the relay improves the throughput gain in a 5.7% in comparison with the case where it is not used recoding.

5.2 Future work

Due to the lack of time during this thesis, it was not possible to make some simulation to back up the conclusion that it was achieved during the analytical evaluation. For this reason, it would be a good starting point for future work to make some simulations based on the different link qualities and to discern when the relay is actually needed in the communication, based on these parameters, and also improving the protocol making possible that the relay turns on or turns off itself depending on the link qualities.

In the developed protocols, it has been set that the only issue able to affect the communication is the distance. In future work it can be added other parameters as fading, to achieve an accurate simulation.

One goal to achieve in future work is to make possible a node to decide if turn on or off itself. For this reason, it has to be decided a threshold for each parameter which affects to communication and specially in some services. For that reason, It is going to be studied in each service which parameters are important and where it would be placed the threshold of them.
The parameters,to take into account, are listed as follow:

- Minimum overhead. This phenomena is considered as a waste of bandwidth, caused by the additional information (control, sequence,etc.) which must be forwarded in addition to data packets in a communication medium. The overhead affects the throughput of a connection.
- Corrupted packets. Interferences and noise can provoke some errors in arrived packets, which leaves as a consequences, an increase in the traffic, in the amount of loss packets, in the overhead and in the throughput.
- Latency is the delay between a packet is through and the packet is received.
- Loss and consecutively loss packets.
- Jitter is produced when the packets from the same source are received with different delays in the destination.
- Throughput is the amount of data per unit time that are delivered through a medium physical or logical, in a network node.

- Channel use. This term is concern to a efficient use of bandwidth.

When the service involved in the communication has relation with streaming, VoIP, online games, or videochat the main parameters to achieve quality of service would be:

- Enough bandwidth.
- Jitter, It should be less than 100 ms.
- The rule of thumb is that no more than 10% of packets should be lost in VOIP networks otherwise the voice quality will be compromised.
- A maximum delay of 150 ms is the rule of thumb for one-way latency to achieve similar quality to POTS voice.

However, for Plane Data transfer this limits can be more flexible.

Bibliography

[311a] ns 3. Api documentation. `http://www.nsnam.org/docs/release/3.13/doxygen/index.html`, 2011.

[311b] ns 3. Wifi. `http://www.nsnam.org/docs/release/3.13/models/html/wifi.html`, 2011.

[8BW10] 802.11s based wireless mesh network (wmn) test-bed. 2010.

[IFN11] Inter-flow network coding for wireless mesh networks. 2011.

[Nag11] Thomas Nagy. The waf book. `http://docs.waf.googlecode.com/git/book_16/single.html#_introduction`, 2011.

[NCO09] Network coding on the gpu. 2009.

[waf11] Waf. `http://code.google.com/p/waf/`, 2011.

Appendix A

How to use NS3 as a library

The goal of this thesis is to introduce Random Linear Coding to NS3. In order to achieve that, two ways can be follow:

- Introduce the new code inside our local NS3, doing necessary make some changes or a new installation of NS3 in the local machine of each new user.
- Or creating a project as a application which will use NS3 as a library, without any modification of NS3.

In this particular case, it has been chosen the second one. It has been create a folder with the Waf code and a wscript where it is specified the path of the libraries which are going to be use for the design. In this way, in the local machine it can be found a folder where is installed NS3, another folder which content a library where the linear network coding C++ codes are stored and a folder where it is placed Waf and wscript and where the applications are built. The last folder, since now will be called Waf folder in this project.

A.1 Creating the Waf folder

[Nag11] In order to create a systems as it is required here, it is needed to download the Waf code. These are the steps to follow:

Firstly, It is needed to download the code:

```
  wget http://waf.googlecode.com/files/waf-1.6.11.tar.bz2
  tar xjvf waf-1.6.11.tar.bz2
  cd waf-1.6.11
  python waf-light
```

The result of this actions will be something like:

```
Configuring the project
'build' finished successfully (0.001s)
Checking for program python              : /usr/bin/python
Checking for python version              : (2, 6, 5, 'final', 0)
'configure' finished successfully (0.176s)
Waf: Entering directory `/waf-1.6.11/build'
[1/1] create_waf:  -> waf
Waf: Leaving directory `/waf-1.6.11/build'
'build' finished successfully (2.050s)
```

Changing the permits of waf folder and execute:

```
  $ chmod 755 waf
  $ ./waf --version
```

The result will be something like:

```
 waf 1.6.11 (a769d6b81b04729804754c4d5214da063779a65)
```

A.2 How to create the wscript

Once the Waf folder is created, the wscript has to specified which will be the behavior of Waf. In this case, these lines will be included:

```
import os 1
APPNAME = 'adhoc_simulations'
VERSION = '1.0.0'
```

How to create the wscript

These lines are in top of the wscript and its behavior is described as:

- 1. To import some functionality of the operative system where the project is taken place.

```
def recurse_helper(ctx, name):
  if not ctx.has_dependency_path(name):
    ctx.fatal('Load a tool to find %s as system dependency' % name)
  else:
    p = ctx.dependency_path(name)
        ctx.recurse(p)
```

The lines above are placed in order to handle the dependencies, and make a warning if it is not possible.

The following line describes the name of a NS3 shared library. Where %s is used to add the correct string which fits in each case.

```
ns3_lib_name = "ns3-dev-\%s-\%s"
```

The wscript is consist of 3 functions; option, configure and build. Option function is where it is configured the options what are required in order to run the program:

```
def options(opt):
  opt.load('toolchain_cxx')
  opt.load('dependency_bundle')
  import waflib.extras.dependency_bundle as bundle
  import waflib.extras.dependency_resolve as resolve
  bundle.add_dependency(opt,
                      resolve.ResolveGitMajorVersion(
                      name = 'gtest',
                      git_repository = 'git://github.com/steinwurf/
                      external-gtest.git',
                      major_version = 1))
  bundle.add_dependency(opt,
                      resolve.ResolveGitMajorVersion(
                      name = 'boost',
                      git_repository = 'git://github.com/steinwurf/
                      external-boost.git',
                      major_version = 1))

  bundle.add_dependency(opt,
                      resolve.ResolveGitMajorVersion(
                      name = 'sak',
                      git_repository = 'git://github.com/steinwurf/
                      sak.git',
                      major_version = 2))

  bundle.add_dependency(opt,
                      resolve.ResolveGitMajorVersion(
                      name = 'fifi',
                      git_repository = 'git://github.com/steinwurf/
                      fifi.git',
                      major_version = 2))

  bundle.add_dependency(opt,
                      resolve.ResolveGitMajorVersion(
                      name = 'kodo',
                      git_repository = 'git://github.com/steinwurf/
                      kodo.git',
                      major_version = 2))
```

The lines above are placed in order to download, in the case that is necessary, the library kodo and its dependencies, where the network coding code is placed. This step is done in order to make easier the use and installation of the project enviroment.

```
   opt.load('compiler_cxx')  1
   opt.add_option('--run',   2
                    help=('Run a locally built program; argument
                           can be a program name,'
                          ' or a command starting with the program name.'),
                    type="string", default='', dest='run')
opt.add_option('--ns3-path', 3
                   help='Install path to ns3',
                      action="store", type="string", default=None,
                   dest='ns3_path')
```

```
opt.add_option('--ns3-type', 4
                    help='The build type used when building ns3
                         [debug|release]',
                       action="store", type="string", default='debug',
                    dest='ns3_type')
```

Where,

1. To look for a C++ compiler.
2. To run a locally built program.
3. To add the path where NS3 is placed.
4. To specify is ns3 is a debug or release (the libraries names have a slightly change depending on that, however this fact may cause some problems.)

To configure waf some of these options have to be specified or the program could not be built. This is an example of that:

```
./waf configure --ns3-path=../../repo/ns-3-allinone/ns-3-dev/
                --ns3-type=debug
```

Where there are specified the NS3 path and the NS3 type in this system. Both values will be changed for each user in his own system, adding his own NS3 path and type.

In the configure function are added the following lines whose behavior is explained below:

```
def configure(conf):
  if not conf.options.ns3_path:    1
    conf.fatal('Please specify a path to ns3')

  ns3_path = os.path.abspath(os.path.expanduser(conf.options.ns3_path)) 2

  if not os.path.isdir(ns3_path):                      3
    conf.fatal('The specified ns3 path "%s" is not a valid '
               'directory' % ns3_path)
```

```
ns3_build = os.path.join(ns3_path, 'build')4
if not os.path.isdir(ns3_build):                5
  conf.fatal('Could not find the ns3 build directory '
             'in "%s"' % ns3_build)
ns3_type = conf.options.ns3_type   6
if not ns3_type:                   7
  conf.fatal('You must specify the build type')
conf.env['NS3_BUILD'] = [ns3_build]         8
conf.env['NS3_TYPE'] = ns3_type             9
conf.check_cxx(header_name = 'ns3/core-module.h',
includes = [ns3_build])     10
conf.check_cxx(lib = ns3_lib_name % ('core', ns3_type),
libpath = [ns3_build])      11
conf.load('toolchain_cxx')                  12
conf.load('dependency_bundle')              13

recurse_helper(conf, 'boost')
recurse_helper(conf, 'gtest')
recurse_helper(conf, 'sak')
recurse_helper(conf, 'fifi')
recurse_helper(conf, 'kodo')
```

where,

1. To warn to the user about adding the NS3 path
2. To add the ns3 path introduced by the user in the sentence waf configure.(An example is shown above)
3. To warn to the user about the NS3 path added in waf configure is incorrect.
4. To add the string build to the NS3 path to go inside the build folder.
5. To warn to the user about the NS3 build folder path added in waf configure is incorrect.
6. To get the type of NS3 introduced by the user in a string variable.
7. To warn to the user about adding the type of NS3 debug or realese.
8. To check header of the function core-modules which is included in the code through the sentence include.

9. To check the library where is included the body of core-modules, and it adds the path where the library is placed.
10. To add the path where the libraries are placed to a environment variable, and use this value in other functions.
11. To add the type of the libraries to a environment variable, and use this value in other functions.
12. To load toolchain_cxx
13. To load dependency_bundle

In the build function are added the following lines whose behavior is explained below:

```
def build(bld):
  bld.load('dependency_bundle')
  recurse_helper(bld, 'boost')
  recurse_helper(bld, 'gtest')
  recurse_helper(bld, 'sak')
  recurse_helper(bld, 'fifi')
  recurse_helper(bld, 'kodo')
  ns3_build = bld.env['NS3_BUILD'] 1
  ns3_type = bld.env['NS3_TYPE']  2
  ns3_libs = [ns3_lib_name % ('internet', ns3_type),
              ns3_lib_name % ('config-store', ns3_type),
               ns3_lib_name % ('core', ns3_type)]   3
  ns3_lib_dir = bld.root.find_dir(ns3_build)  4
  ns3_libs = ns3_lib_dir.ant_glob('*.so')  5
ns3_libs = [str(lib).split('/')[-1] for lib in ns3_libs]  6
  ns3_libs = [lib[3:] for lib in ns3_libs]  7
  ns3_libs = [lib[:-3] for lib in ns3_libs]  8
  bld.program(source = ['adhoc.cc', 'pep-wifi-helper.cc',
                        'code-header.cc', 'pep-wifi-net-device.cc'], 9
              target = 'adhoc',
              libpath = ns3_build,
              rpath = ns3_build,
              includes = ns3_build,
            defines = ["NS3_LOG_ENABLE"],                    10
              lib = ns3_libs,
              cxxflags = bld.toolchain_cxx_flags(),
              features = 'cxx cxxprogram',
              use = ['kodo_includes', 'boost_includes',
                     'fifi_includes', 'sak_includes'])
```

where,

1. To extract the path where are placed the libraries from the environment variable.
2. To extract the type of the libraries from the environment variable.
3. To create a list with the libraries used for a code.
4. To go from the root to the path where the libraries are placed.
5. To create a list with all the libraries placed in this folder.
6. To cut the name of the library, as this example:
   from : //path/to/libraries/build/libns3-dev-wimax-debug.so
   to: libns3-dev-wimax-debug.so
7. To cut the three letters from the beginning: now, the name will be something like ns3-dev-wimax-debug.so.
8. To cut the three letters from the end: now, the name will be something like ns3-dev-wimax-debug.
9. To declare a program to compile and built.
10. To enable NS3 LOG.