ET4394 - Wireless Networking
Ns3 Project

Giannopoulos Efthymios, 4490827

April 29, 2016

Abstract

This report depicts the work that was done for the ns3 project of the Wireless Networking course. The report is organized as follows.
First is the introduction where a brief description of the simulated scenarios is given. Second is the implementation section, where in more detail the implementation of the scenarios is presented. Third is the results section where the produced graphs and results are discussed. Finally there is the conclusion, alongside the references and appendix.

Introduction

IEEE 802.11b uses the 2.4GHz band and extends throughput up to 11 Mbit/s. It uses the CSMA/CA method, which means that nodes listen the channel and transmit when it is idle. Collision detection cannot occur, therefore if a collision happens, the node that has not received an ACK after some time transmits again. In addition, IEEE 802.11b devices suffer interference from other products operating in the 2.4GHz band. The maximum raw data rate of 11 Mbit/s is not achieved in practice as propagation loss, interference and CSMA/CA protocol overhead put a bound. [1]

In this project the Wifi IEEE 802.11b is simulated using the infrastructure topology, meaning one access point and different number of station nodes. Different parameters of the wifi network are changed in order to obtain the different results in network performance.

During this project two main scenarios are implemented:

- The first scenario is about the critical distance from the access point. The idea is to calculate at which distance will the communication with the access point break and how is that distance influenced by modifying the parameters of Receiver Gain (RxGain) and Receiver Noise Figure (RxNoiseFigure).

- The second scenario is about calculating the throughput in the network. Basically, throughput is measured as the total successfully received bits

during the time space of interest. Successfully received packets (thus bits too) at every node are captured both at physical and MAC layer in order to calculate the throughput at these different layers. Application layer packet size and number of nodes are the parameters that are changing per simulation.

In the following sections the implementation of the above mentioned scenarios is presented as well as the obtained results.

Implementation

"Ns-3 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" as it is stated in the ns3 homepage. The wifi network is simulated using ns3, as it supports various parameters to modify in the physical, data-link and network layer. The classes that are used are programmed in C++. The ns3 version used is ns-3.24.[2]

First Scenario

As previously mentioned, the first simulated scenario is about calculating the critical distance from access point. Distance from access point, RxGain and RxNoise Figure are changing values per simulation in order to obtain the different results. Throughput is measured every time until it reaches the zero value which means that there is no traffic, so the access point and the nodes are disconnected.

Firstly, the nodes that consist the network need to be created. In this scenario, using the NodeContainer class, one access point and two nodes are created. The second step is to select the Wifi standard. Using the WifiHelper class, the standard is set to 802.11b. The control rate algorithm is chosen to be the Constant Rate Algorithm, and the data rate is set to 11Mbps, which is the higher rate for 802.11b. Furthermore, the physical layer is created using the YansWiFiHelper class. Error rate model, RxGain and RxNoise Figure are the three parameters of this class that are modified. The error rate model is set to YansErrorRateModel, RxGain has values of 0dB, 5dB or 10dB per simulation and RxNoise Figure takes values of 0dB, 10dB or 20dB. The YansWifiChannelHelper class is used to create the channel model. The propagation delay is set to Constant Speed Propagation and the propagation loss is set to the log distance propagation loss, where gamma variable has a value of 3. Next step is the Wifi Mac parameters. A default Nqos Mac is used. The devices are finally installed at both the access point and the station nodes using the NetDeviceContainer class.

Another important information that should be set is the topology of the network. The MobilityHelper class is used in order to create mobile or stationary nodes and it is also used to specify their starting or permanent positions. In this scenario the access point is placed at the (0,0,0) position. The two station nodes are placed along the y axis symmetrically from the access point. In each simulation, their position changes in the value range (0-265) meters alongside the y-axis. The InternetStackHelper class is used then in order to install the Internet Stack on all nodes. The Ipv4AddressHelper class is used in order to assign the IP adresses of each node.

One station node executes the OnOff application, introduced by the OnOffHelper class. It is supposed that the node generates traffic intended for the other station node. The node generates traffic for a time fraction then stops and then gener-

ates again and so on, according to an on-off pattern. In this case the off pattern is set to zero. Consequently, the station node generates traffic continuously. The type of traffic is UDP, the packet size is set to 1024 bytes. The other station node executes the Packet Sink application, introduced by the PacketSinkHelper class. Basically, the node just receives packets that are intended for it.

Finally using the FlowMonitorHelper, the throughput for the flow $Station1 -> Station2$ is calculated. Thus, when it reaches zero, it means that in that distance there is no connection with the access point.

Second Scenario

In this scenario, physical layer throughput as well as mac layer throughput are calculated. Throughput is measured while the number of nodes and the transmitted packet size changes. The measurements are done for a static network.

In order to simulate this scenario, there were some changes in the previously mentioned code. This time the network topology consists again of one access point, which is fixed at the position (0,0,0). The number of station nodes varies from 2 to 30 using a step of 2 in each simulation. Figure 1 depicts the network for 30 station nodes. The idea is that odd station nodes using the on-off application, generate continuously traffic for the even station nodes which use the packet sink application. For example, when station nodes are two: 1 sends to 2, when station nodes are four: 1 sends to 2 and 3 sends to 4.

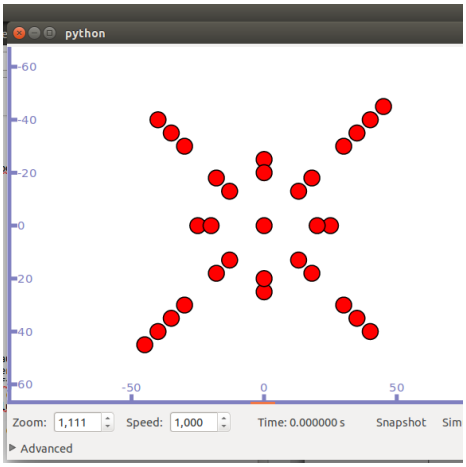

Figure 1: Wifi Topology Consisting of 30 station nodes - The central node is the access point

Different than before, the packet size is no longer set to 1024 bytes but varies in the values of 256 bytes, 512 bytes and 1024 bytes. In addition, RxNoiseFigure and RxGain are set to 0dB and are no longer varying per simulation.

Finally the flow monitor is not used in this code. Instead, in every node of

the network, traces of successfully received packets in physical and mac layer are installed. Specifically, when a node receives successfully a packet, either in the physical layer or in the mac layer the corresponding function fires and stores those successfully received bytes. This information is used to retrieve the average throughput in the network. In the next section, results are presented and discussed.

Results

First Scenario

Receiver noise figure is the difference of SNR (in dB) between the input of an amplifier and its output. Electronic components introduce thermal noise which is unavoidable. In addition, practical components introduce additional noise. Therefore, the SNR observed in input will be higher than that observed in output. Ideally, RxNoise Figure should be 0 dB.

In this scenario, for different RxGain (equal to 0dB, 5dB and 10dB), different RxNoise Figure is also set with values (0dB, 10dB, 20dB). In all those cases, in each simulation all nodes are steadily moved away from the access point, until the distance, where throughput reaches the value of 0 is reached. Each case is simulated three times using randomness in each simulation. Thus the mean value is obtained and is used in the graphs. The produced graphs are these shown below.

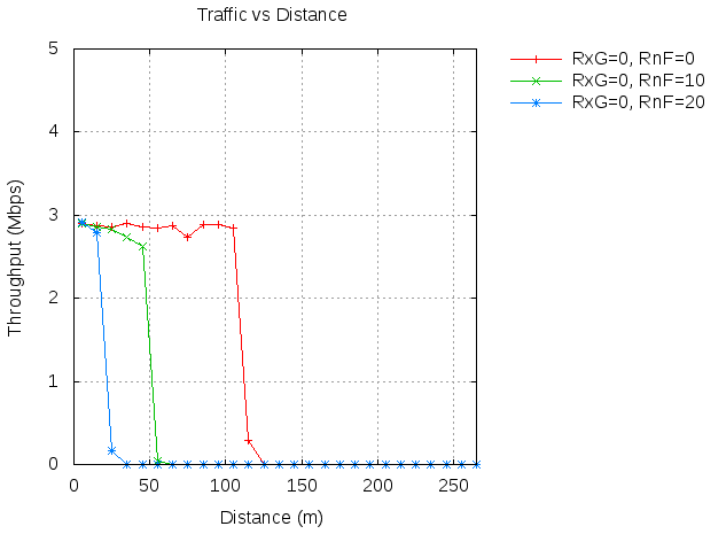

Figure 2: Traffic vs Distance (1)

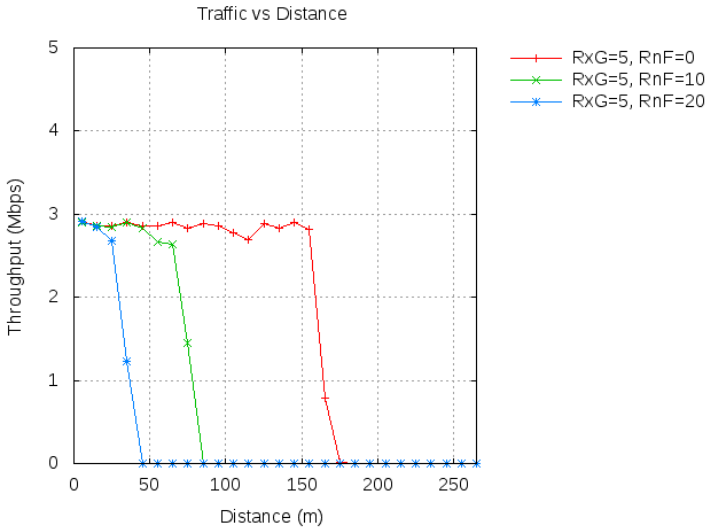

Figure 3: Traffic vs Distance (2)

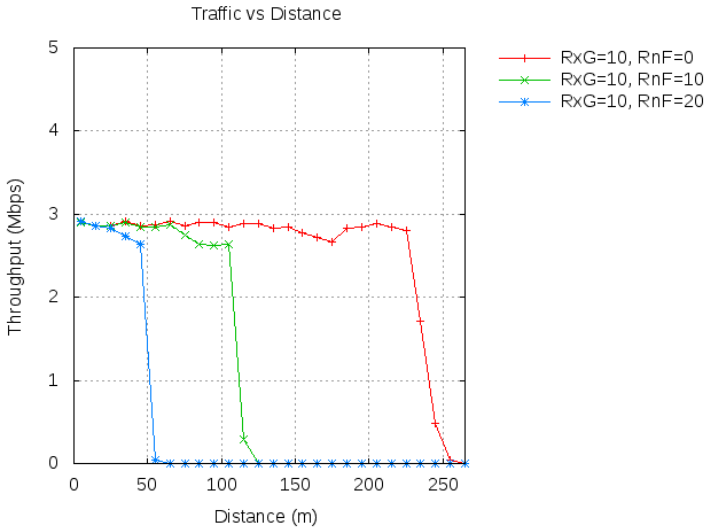

Figure 4: Traffic vs Distance (3)

It is observed that in every RxGain case a 10dB increase in receiver noise figure results to halving down the critical distance. On the other hand for certain value of RxNoise Figure, an increase in RxGain leads to an increase of the critical distance. However, how much is the increase, depends on the certain

value of RxNoise Figure. For example in case that RxNF=0dB an increase in RxGain from 0dB to 5dB would move the critical distance by 50 meters, whereas when RxNF=10dB an increase in RxGain from 0dB to 5dB, would move the critical distance by 30 meters approximately.

Second Scenario

In this scenario, nodes are steadily increased per simulation starting from 2 nodes and ending at 30 nodes. Each time the simulation is run three times with randomness introduced. For each case of nodes three different packet sizes are set (256 bytes, 512 bytes, 1024 bytes).

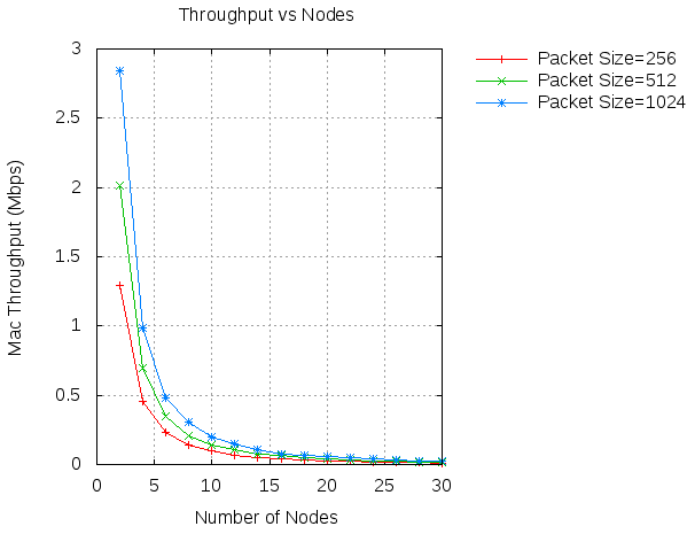

Figure 5: Mac Throughput

As far as the MAC Throughput is concerned, it is observed that as the number of nodes increases the throughput decreases exponentially. It is also observed that for the same number of nodes the higher is the packet size the better is the throughput, especially in the case where few nodes are present.

We set MAC throughput as the successfully received MAC packets of nodes (thus bits as well) during the time space of 9.5 seconds (that applications run). Successfully received MAC packet, means that a packet is sent to a node, the MAC header is read, the node knows that the packet is meant for it, it strips off the header and gets the data in order to forward them to Network layer.

Since half of the nodes are the senders and the other half of them are the receivers, it should be expected that for the time that the applications run, the successfully received MAC packets, should occur at half of the nodes. That is also confirmed by printing the total rx MAC packets by each node. Those

having the on-off application installed, never receive successfully a MAC packet, as no packet is sent to them.

Therefore, it is observed that as the nodes increase the total bandwidth needs to be shared, thus the throughput is decreasing exponentially.

On the other hand the Physical layer throughput produced whole different curves. The topology discussed in the previous section is used. Therefore, all nodes are inside the transmission range of every other node. This means that everyone listens everyone. Thus, all nodes successfully receive physical layer packets, that they forward to MAC layer, where they are dropped in the cases that they are not intended for the specific node. Since this is the case it is observed that as the number of nodes increases the throughput also increases. Transmissions from every node reach each other node and these packets are successfully received in the physical layer.

There is an upper bound. After this bound the throughput starts decreasing again. As the number of nodes is increased over a limit, there is too much traffic inside the network which leads to more collisions. Consequently, the successfully received packets at the physical layer decreases, so does the throughput. It should also be noted that for bigger packet sizes again it is observed better throughput.

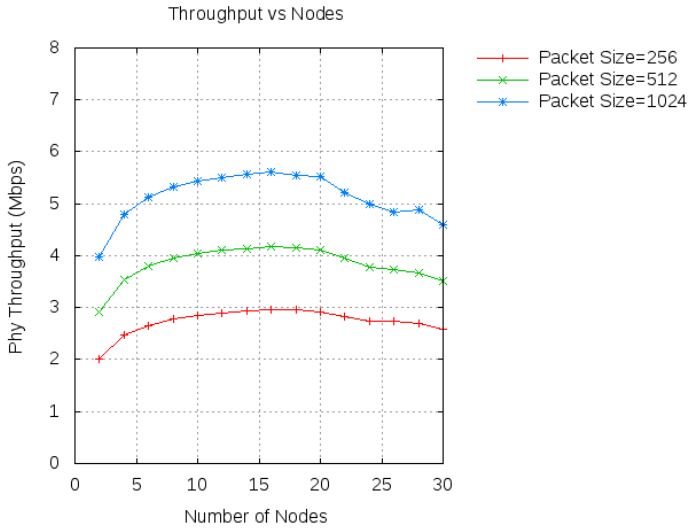

Figure 6: Phy Throughput

Conclusion

It is concluded that:

- Increasing the RxNoise Figure by 10 dB can lead for fixed value of RxGain, to halving down the critical distance
- Increasing the RxGain for fixed RxNoise Figure values can increase the critical distance. How much the distance is increased depends on the RxNF value.
- MAC throughput decreases exponentially as the number of nodes increases.
- Phy throughput increases for a certain number of nodes and then starts decreasing due to collisions because of the high traffic.
- In both throughput cases bigger packet size contributes to higher throughput.
- It is also confirmed that the raw throughput of 11 Mbps introduced by IEEE 802.11b is not achieved.

Appendix

In this section the code written for all the scenarios as well as the scripts for producing the results are presented.

The graphs are produced by the tool gnuplot. In order for the graps to be produced in every machine gnuplot should be installed. This is done by executing the following commands:

```
1 sudo apt-get update
2 sudo apt-get install gnuplot
```

For the first scenario the scripts "dist.sh", "0.pd", "5.pd", "10.pd" as well as the "read" executable program, need to be placed inside the working ns3 directory (/home/source/ns-3.24). In addition the source code for the first scenario named "wifi0.cc" needs to be placed in the scratch directory.

Finally running the command inside the ns3 working directory:

```
sh ./dist.sh
```

will execute all simulations and scripts and finally will produce the three graphs that are also shown in the report. This procedure is expected to last from fourty minutes to one hour!

For the second scenario the scripts "nd_ps.sh", "mac.pd", "phy.pd" as well as the "read1" executable program, need to be placed inside the working ns3 directory (/home/source/ns-3.24). In addition the source code for the second scenario named "wifi1.cc" needs to be placed in the scratch directory.

Finally running the command inside the ns3 working directory:

```
sh ./nd_ps.sh
```

will execute all simulations and scripts and finally will produce the three graphs that are also shown in the report. This procedure is expected to last for long time as well!

All the code used for this project can be found on github `https://github.com/makbut/et4394_ns3_project`. It should be noted that the code is not totally mine. As browsing through the [3, 4, 5] many code can be found. All these piece of codes were modified or used as a whole by me, by also inserting mine ideas in certain places.

References

[1] Wikipedia. IEEE 802.11b. https://en.wikipedia.org/wiki/IEEE_802.11b-1999.

[2] Ns3 homepage. https://www.nsnam.org/.

[3] Ns3. Wifi Module in Ns3. https://www.nsnam.org/tutorials/consortium14/ns-3-training-session-5.pdf.

[4] Ns3. Ns3 Model library. https://www.nsnam.org/docs/models/html/index.html.

[5] Ns3 users google group. https://groups.google.com/forum/#!forum/ns-3-users.