EE 595 Experiment 0

First Wireless Experiment

I. Goals

Students will use a provided ns-3 program to plot the performance of a simple wireless channel as different parameters are varied.

Fig. 1: Topology of first wireless experiment

Students will submit a document (brief report) with plots that they have generated and answers to the questions below.

A. Introduction

We will use the provided `link-performance.cc` program to experiment with varying configuration parameters and their effect on packet receptions over a simple wireless link.

The physical layer model is the abstraction representing the actual 'physical layer' (the layer that carries source/information bits over a point-to-point link) within a network simulator. Refer to Fig. 2, that shows a *simplified* link-layer model with the core elements - i. channel encoder, ii. channel modulator, iii. additive noise channel, iv. demodulator and v. decoder.

In reality, link simulation is considerably more complex than the simplified model abstraction above - largely due to 2 components:

i. many channels have non-additive noise, and include multipath propagation between the transmitter

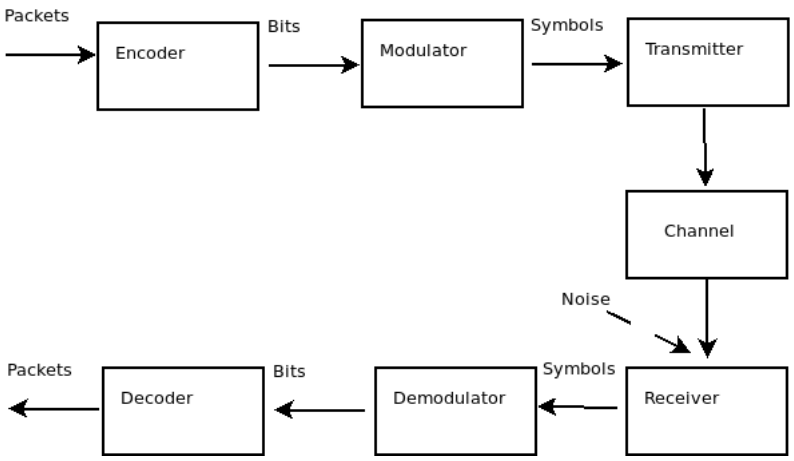

Fig. 2: Simplified PHY model

and receiver (the complexity of which increases with modern day larger bandwidth, Multi-input/Multi-output (MIMO) systems) and ii. incorporation of the RF/mixed signal elements and the digital baseband decoder. A high-fidelity 'link simulator' in industry requires considerable effort to set up and significant computational (algorithmic) skills to conduct end-to-end link simulation. In summary, need for accurate link-layers lead to *long* simulation times, and therefore, simulator construction for the physical layer usually involves trade-offs (accuracy/complexity vs run time). The goal of this exercise is to introduce some elements of this process of abstraction and the tradeoffs involved.

The goal of a reliably designed communications link is to achieve (near) error-free communications between the two end points. At the physical layer, the performance of any such link design is characterized by the bit error rate (BER) curve vs. the Signal-to-Noise Ratio (SNR) at the receiver input (as shown in Fig. 2). Hence, we will convert the 'real' system shown in Figure 2 by its model equivalent where the entire end-to-end link is summarized by the transmit power, noise power, path loss model, frequency, and distance, to arrive at a notional SNR. Hence, given a PER vs SNR curve, the receiver accepts/rejects a transmitted packet as a Bernoulli coin toss with $p = PER$.

As has already been seen from the channel (path loss) models, the received signal power is a function of distance and frequency. Thus in general, the PER vs SNR function depends on some key system parameters: a) link distance and frequency, b) transmit power and receiver noise power, c) Modulation and Coding Scheme (MCS) employed and d) packet length.

The goal of Experiment 0 is to walk you through such a process flow and develop a packet level abstraction for use in ns-3.

For simplicity, initially we use uncoded binary phase shift keying (BPSK) modulation, with no channel encoding, over an additive white Gaussian Noise (AWGN) channel model with a deterministic path loss (channel) model given by the Friis (free-space propagation) equation:

$$P_r = P_t\Big(\frac{\lambda}{4\pi d}\Big)^{\alpha} \tag{1}$$

using standard notation for the received power P_r as a function of λ the signal wavelength, d the distance between the transmitter and receiver, and α the path loss exponent. For free space, the path loss exponent is 2. For an Additive White Gaussian Noise channel (AWGN) the receive power can be converted to receive SNR γ (at the input to the receiver, see Fig. 2) according to:

$$\gamma = \frac{P_r}{N_0 B} \tag{2}$$

where $N_0 B$ is the noise power in the (one-sided) signal bandwidth B. To compute the BER as a function of the link SNR (measured at the input to the receiver), we use the following well-known result for BER for uncoded BPSK (see any text such as [] or online source)

$$BER = Q(\sqrt{2\gamma}) \tag{3}$$

where Q($\cdot$) is the tail of a standard (zero-mean, unit variance) Gaussian random variable, i.e. $Q(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{u^2}{2}}\, du$. We can convert the above BER to an equivalent PER for the case of uncoded modulation, as follows:

$$PER = 1 - (1 - BER)^L \tag{4}$$

where L is the length of the packet data or payload.

B. Evaluation

Consider a model for BPSK communications link over which packets of various sizes are sent. By varying the following parameters:

- **Frequency**
- **Distance**
- **Packet size**
- **Transmitter signal power**

- **Receiver noise power**

We can arrive at a specific probability of packet error (the **Packet Error Ratio**, or PER). If we now send a large number of packets through the system, then on average, a fraction equal to PER of packets sent will be lost on the channel and dropped at the receiver.

In the ns-3 code we will make use of the Friis deterministic path loss model: Friis path loss.

Note: in a future assignment, we will vary the modulation and coding, and the path loss model.

```
$ ./waf --run 'link-performance --PrintHelp'
ns3.29-link-performance-debug [Program Options] [General Arguments]

Program Options:
    --distance:    the distance between the two nodes [25]
    --maxPackets:  the number of packets to send [1000]
    --packetSize:  packet size in bytes [1024]
    --transmitPower:  transmit power in dBm [16]
    --noisePower:  noise power in dBm [-100]
    --frequency:   frequency in Hz [5e+09]
    --metadata:    metadata about experiment run []

General Arguments:
    --PrintGlobals:              Print the list of globals.
    --PrintGroups:               Print the list of groups.
    --PrintGroup=[group]:        Print all TypeIds of group.
    --PrintTypeIds:              Print all TypeIds.
    --PrintAttributes=[typeid]:  Print all attributes of typeid.
    --PrintHelp:                 Print this help message.
```

This output shows that there are a number of custom program options as well as default values. For instance, the distance between the two nodes is 25 meters by default, but can be changed. To summarize,

the program will, by default, run a network simulation that sends 1000 packets, each of size 1024 bytes, from a transmitter to a receiver. The transmit power by default is 16 dBm (16 dB in reference to 1 mW of power), and the noise power is -100 dBm. The signals are sent at a notional frequency of 5 GHz.

Try running with defaults (i.e. by not specifying any arguments), as follows:

```
$ ./waf --run 'link-performance'
Waf: Entering directory `/ns-3-ee-595/build'
Waf: Leaving directory `ns-3-ee-595/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (3.888s)
sent 1000 rcv 1000 drop 0 per 0 error 0
```

The program output shows that 1000 packets were sent, 1000 were received, 0 were dropped, for an overall packet error ratio (per) of 0. Please ignore the fifth value (error) for now; we will cover this next week.

There are also two output files saved. The first one is `link-performance-rssi.dat`. This file records a timestamp and a received signal strength (in dBm) for every packet sent. Let's look at a sample of this file:

```
$ head link-performance-rssi.dat
1.10008 -58.386
1.20008 -58.386
1.30008 -58.386
1.40008 -58.386
1.50008 -58.386
1.60008 -58.386
1.70008 -58.386
1.80008 -58.386
1.90008 -58.386
2.00008 -58.386
```

Each packet is received 100 ms after the previous one, and each has an RSSI of -58.386 dBm (there is no variability). So, the sending of the signal from the transmitter to the receiver dropped the signal strength from 16 dBm to -58 dBm, due to the free space path loss at this frequency.

The second file is `link-performance-summary.dat`; let's have a look:

```
$ cat link-performance-summary.dat
1000 1000 0 0 0
```

These are the same five values we saw above, without the annotations. Note that each time the simulation is run, this file will be appended, rather than overwritten, allowing one to manually create a data file for plotting based on the command-line simulation runs.

This communications link is too solid (too high of an SNR) to be of much interest. There are a number of possible ways to degrade the performance, such as increasing the distance between the sender and receiver (which will, in turn, cause the receive power to drop, and correspondingly the SNR to drop). Try:

```
$ ./waf --run 'link-performance --transmitPower=0 \
--noisePower=-90 --distance=100'
sent 1000 rcv 0 drop 1000 per 1 error 0
```

Now we observe that no packets are getting through. We can also check the link-performance-summary.dat file again:

```
1000 1000 0 0 0
1000 0 1000 1 0
```

In this manner, you can use the simulator to build up your own data file, and plot relationships between the parameters of interest (e.g. the PER vs. distance). However, this may be tedious, and could be scripted. We have written a sample script to automate the running of this program across a range of values, and then to take the resulting data file and generate a plot. The script is written in Bash, and is found in the file `contrib/simple-wireless/experiments/link-performance/run-link-performance.sh`. By default, the program does the following:

1) Sets some parameters to non-default values, and steps through a range of distance values, recording the observed link performance summary in a file.
2) Uses Python matplotlib to plot the PER vs. distance relationship.
3) Creates a timestamped results/ directory, and copies the data files, plot, and script to the results directory.

Try running this script and looking at the PDF plot in the results directory. It should look like the following figure:

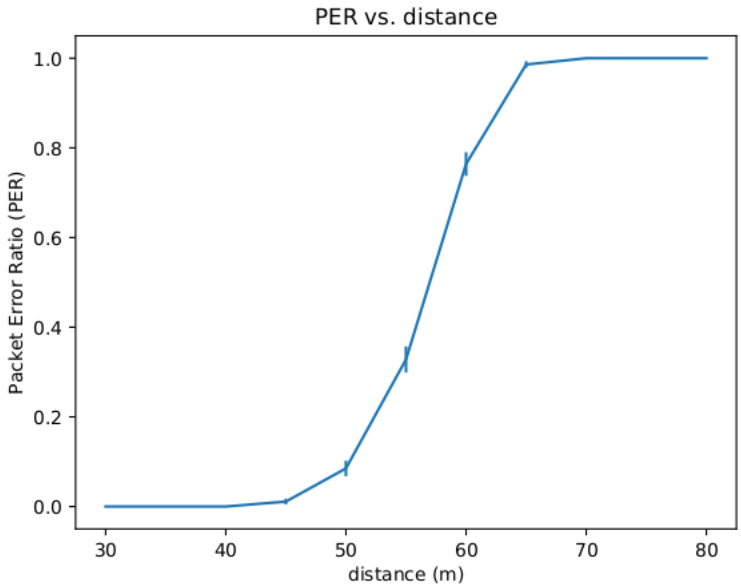

Fig. 3: link-performance.cc PER vs distance

This script may be helpful to automate the running of ns-3 to answer some of the questions below (the script will need some modifications. However, students are free to write their own script or use their own plotting utility of choice.

C. Questions

1) What effect on link performance does packet size have? For a given PER, how much greater or lesser distance can be supported if the packet size is 100 rather than 1000 bytes?
2) How strong of an effect does frequency have on results? 2.4 GHz is another popular frequency for unlicensed operation. How much more or less range can be supported for the same PER if the frequency is changed to 2.4 GHz?

3) Instead of plotting PER vs. distance, provide a comparable plot of PER vs. transmit power (stepping through a range of transmit power so that the PER varies from 0 to 1, and with enough data points in the interesting region of the graph to show a curve). Produce a plot similar to the PER vs. distance plot above.