Data Communication Networks

HW 2: Data Link Layer Simulation With Python

Dr. Mohammadreza Pakravan

Note: You can only use python language to implement the questions. You are free to use the third-party modules. You can change the template codes, but it is not recommended.

1 ARQ

ARQ systems use coding to make a certain level of errors detectable at the receiver. In the case of detecting error occurrence in the received frame, the receiver requests the transmitter to resend the frame. As a block diagram, the system that you are going to implement here is depicted in Fig. 1.

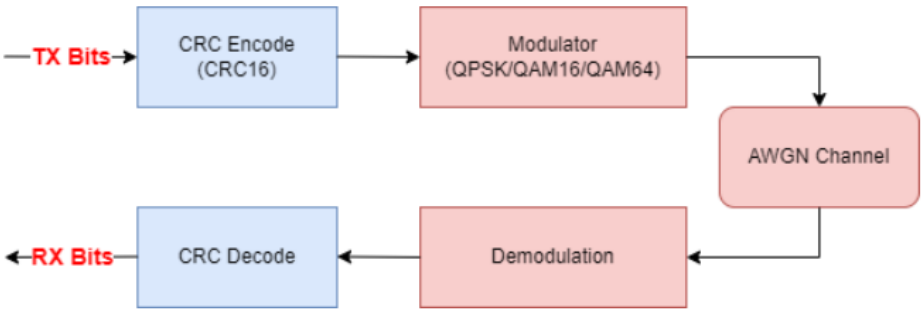

Figure 1: A simple block diagram of the data transmission system used in the ARQ section of this assignment.

A recommended Python library for carrying out this section: komm
Python notebook file for writing the code of this section: `ARQ.ipynb`

Questions

1.1

We first try to examine the effect of different channel conditions on the bit error rate, so there is no need to use any coding and ARQ procedures for this question. Create 10 random frames, every 1000 bits. Transmit these frames through an AWGN channel using QPSK modulation (or equivalently, QAM4). You should sweep SNR values from -5 dB to 20 dB. You may choose the number of points in this interval as you see fit. It is also recommended that you run each experiment a few times to get smoother results.

You must deliver the following:

- A plot of bit error rate (BER) versus SNR, both in dB.
- A plot of channel throughput versus SNR, both in dB.

Explain the behavior of each curve. Do the ultimate values of curves make sense? Please bear in mind that channel throughput is defined as "the number of bits that have been successfully transmitted through the channel per unit time", you may safely assume that you have enough bandwidth and full transmission of each frame takes exactly one second if no retransmission is needed.

1.2

Using the BER curve that you found in the previous question, find the minimum SNR value that we need, so that with a probability of at least 90%, we can transmit a frame with at most 2 retransmissions.

1.3

Implement ARQ, this time restrict yourself to a 4 dB interval (2 dB at each side) around the SNR value that you found in the question 1.2. To detect transmission error, you must append CRC to your frame before sending it, the receiver may only rely on the outcome of CRC decode to decide whether or not the frame has an error.

You must deliver the following:

- A plot of the total number of bits that were sent (regardless of whether or not they were correctly received, this only concerns the transmitter's viewpoint) in dB versus SNR, also in dB.

Is this system stable? In other words, does a small decrease in SNR result in a rather small loss of efficiency?

2 Mini NS-3

In this question, we want to create a mini NS3. You must implement only data link (DL) layer algorithms. Other layers have been implemented.

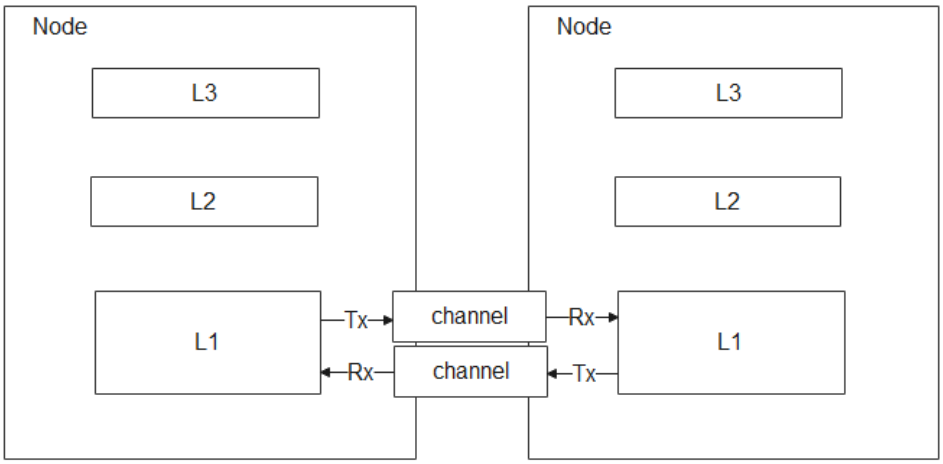

Figure 2: A simple Network setup for simulation.

According to Fig. 2, the network has two nodes. For the sake of simplicity, all layers upper than L3 are abstracted in L3. This layer task is only generating random packets in Tx mode and counting received packets in Rx mode. Layer 1 tasks are simplified: Adding CRC to the L2 frame and injecting bits (without any modulation) into the channel in Tx mode and reversing these operations in Rx mode. For the sake of simplicity assume that L1 can recover frames ideally. Layer 1 has two connected channels, one is for Tx and another is for Rx. These 2 channels (Tx and Rx) may have different characteristics. To ease implementation and reduce the simulation time assume that channels are binary symmetric channels (BSC). Also, channels have transmission and propagation delays.

In the given template codes, each layer is implemented as a class. After creating (instancing) each layer you should bind them to its node. Each layer interacts with its upper/lower layers indirectly i.e. the layer asks the node to call the target layer (upper/lower). In this procedure, the node acts as a proxy. For the next step, all objects (2 nodes, 2 channels) must bind to the simulator object.

We want an event-driven simulator. Simulators like ODE solver (like the Euler method) always take the same time step and do an action corresponding to conditions (change y based on derivation). It means uniform time sampling. But in event-driven simulators, time progresses according to available events (interrupts) and it may be nonuniform time sampling. For example, there are 2 events in the event queue: events "A" and "B". Event A has been set 2 seconds later and event B is placed 3 seconds later. The simulator goes to the nearest event which is event A and lets that event be handled. Assume that event handling creates a new event "C" at 3 seconds later. The event queue now has 2 events: event B at 1 second later and event C at 3 seconds later. Now, the simulator decides to handle event B, and the time updates to 1 second later.

In the template codes, each network object (node, channel) has an independent event queue. The simulator inspects each queue every time (after an event is handled completely) and selects the nearest event across all queues and handles that event. You must only design these event handlers. The defined event format is a tuple of timestamp, event type, args (like a packet or sequence number): `(time, type, *args)`

Four event types are defined: 1- **ch transmit**: Rx receives the noisy packet and parses it. 2- **DL timeout**: the sender timer timeouts. 3-**DL start**: L2 intends to transmit a new frame and pass it to L1. 4- **ch cancel**: cancel packet transmitting i.e. delete *ch transmit* event contains special packet. This event type (ch cancel) is not mandatory and you can skip using this event type.

Questions

2.1 How does the simulator work? How each object in the code behaves?

2.2 Complete Go Back N, Selective Repeat algorithm in DataLink.py file.

Answer next questions in `DL_simul.ipynb` notebook. Calculate and plot link utilization for all 3 data link algorithms. The default paramters: packet size=96, transmission rate=10^4, CRC divisor= $0xB$, $BER = 10^{-4}$, $a = 1$.

2.3 Sweep parameter 'a' between 0.01 to 100 and calculate link utilization.

2.4 Sweep packet size between 50 to 1000 and calculate link utilization. Does utilization changes? Why? Parameter 'a'=1.

2.5 Repeat question 3 while the ACK channel is lossless (BER=0). Does the utilization change? How to achieve a ideal channel for ACK?

What Should I prepare?

You must upload the simulation python file and notebooks. Also, prepare a report file in pdf format. Compress all files and rename the compressed file to `DN_HW2_StudentID.zip`.