Lab 4: Reliable Transport (10 Points)

1 Download Lab

Download the Lab 4 files from Brightspace. The source code will be inside the directory "`Lab4/`". You must use the environment from Lab 0 to run and test your code. Next, open a terminal and "`cd`" into the "`Lab4/`" directory. Now you are ready to run the lab!

2 Simulated Network Configuration

For this lab, you will work with a simplified version of the network simulator used in Lab 3. The simulated network's configuration is specified in the file "`01.json`". The network comprises two clients A and B connected using the router "1" (Figure 1). Client A sends a file to client B via the router. The latency of each link is 1 second. The Maximum Segment Size (MSS) is set to 256 bytes, i.e., you cannot send a packet with payload larger than 256 bytes. The router can drop packets (both data and acknowledgment packets) randomly based on a loss probability parameter. Assuming a loss probability value of p, on receiving a packet, the router will drop that packet with probability p. The router will never re-order the received packets.

We will use the same network configuration as above throughout this lab and for the evaluation.

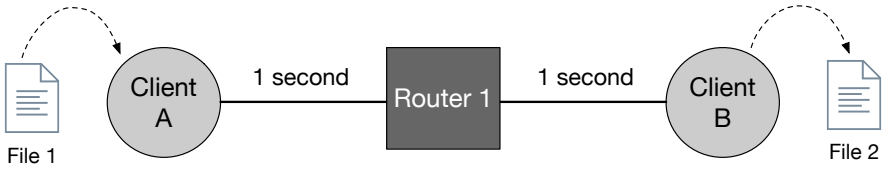

Figure 1: Simulated network topology. Client A sends the contents of File 1 to client B via router 1. Client B writes the received bytes into File 2.

3 Running the Code

You must run and test your code on `eceprog` using the environment from Lab 0. If your code does not run in that environment, you will not get any credit!

Start the simulator using the command,

```
$ python3 network.py 01.json [send file path] [recv file path] [loss probability]
```

The "`01.json`" file argument specifies the configuration of the simulated network, as described in the above section. The "`send file path`" and "`recv file path`" arguments specify respectively the file to be sent over the network and the file to which the received bytes are to be written at the receiver. The "`loss probability`" argument specifies the probability of dropping a received packet at the router. **The loss probability value (p) must be an integer in the range $0 \leq p \leq 99$.**

4 Task

The task is to send a ***single*** ASCII text file ***reliably*** from client A (sender) to client B (receiver) via the router. The received file must be identical to the sent file, even under packet loss at the router.

The provided code in the "`myClient.py`" file already implements the file transfer logic **assuming no packet loss.** The code first sets up a connection using `SYN, SYN-ACK, ACK` packets. After the connection is established, client *A* reads and sends the data from "`sendFile`" to the router by creating a packet using the "`Packet`" class in "`packet.py`" file. The router forwards the received packets to client *B*. Once all the data in the "`sendFile`" has been sent, the code terminates the connection using `FIN, FIN-ACK, ACK` packets. **Connection set up and termination packets are never dropped** in this simulator. Client *B*, on receiving a data packet from the router, writes the payload of the packet to the "`recvFile`". You may run the provided code to check its behavior using the following command,

```
$ python3 network.py 01.json sendfiles/file4.txt recvfiles/file4.txt 0
```

Your task is to augment the provided code in "`myClient.py`" with a **reliability protocol** that ensures reliable file transfer **even under packet loss.** You are free to implement ***any*** reliability protocol of your choice. We discussed several different options in Lecture 22 (Transport Layer I). Further, please refer to section 6 for grading considerations for choosing the right protocol. You will do all your implementation inside "`myClient.py`". **You must not modify any other file in the** "`Lab4/`" **directory**.

5 Output

The simulator prints different characters on the terminal to track the progress of the file transfer.

`+` is printed every time the router receives a packet from client *A* and forwards it to client *B*.

`[+]` is printed every time the router receives a packet from client *A* and drops it.

`@` is printed every time the router receives a packet from client *B* and forwards it to client *A*.

`[@]` is printed every time the router receives a packet from client *B* and drops it.

Once the connection successfully terminates, the simulator prints the file transfer statistics on the terminal. This includes the total time of transfer and the total bytes sent (including the bytes that were dropped). At the end, the simulator matches the received file against the sent file, and prints either "SUCCESS" or "FAILURE" depending upon whether the sent and received files are identical or not. **If you print any custom / debug statements to the terminal, it will result in a 20% grade penalty.**

For debugging purposes, the simulator also generates `.dump` files inside the "`logs/`" directory, for both the clients and the router. The files contain the log of each received packet with its header and payload contents. If a packet received at the router was dropped, the log also indicates that.

6 Grading

We will test your code using various test cases with different loss probability values. For each test case, if the received file is not identical to the sent file, you will receive 0 points. If the received file is indeed identical to the sent file for a test case, you will receive 70% credit. You will receive the remaining credits for that test case if for your solution, the **number of bytes sent** is within 1.5× (for 15% credit) or within 1.75× (for 10% credit) or within 2× (for 5% credit) and the **file transfer time** is within 2× (for 15% credit) or within 3× (for 10% credit) or within 4× (for 5% credit) of a "good" student solution (e.g., a correct solution that is in the 90+ percentile for both the performance metrics). Your final grade will be the sum total of points obtained across all the test cases.

7 Submission

You are required to submit **one** file "`myClient.py`" on Brightspace. **Do not submit a .zip file.**