Computer Networks **Instructor: Dr. Deepak Tosh**
Spring 2022 **Due Date:** 03/13/2022

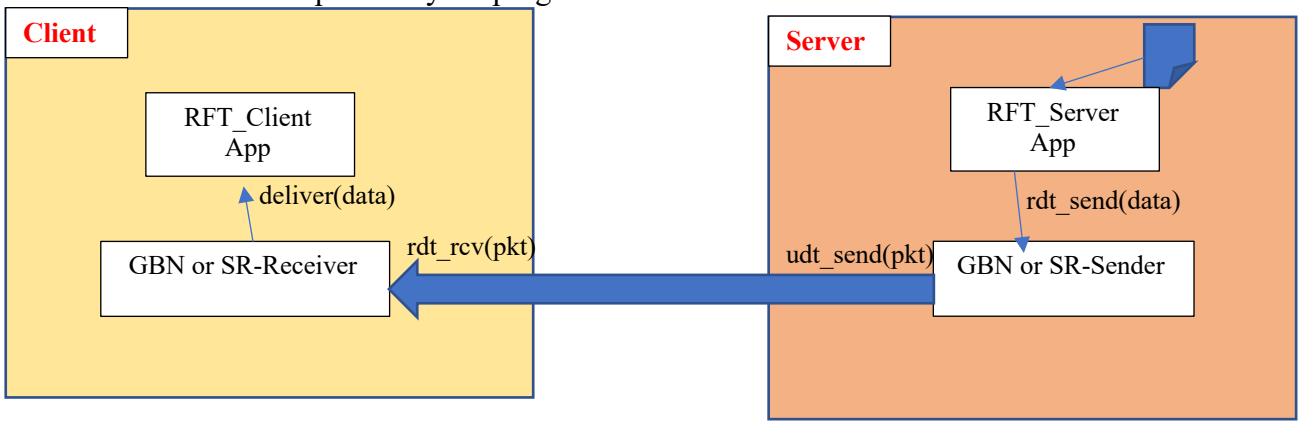
Assignment # 2b: Reliable File Transfer

In this assignment, your goal is to write modify your already-developed file transfer server and client to communicate using the **Pipelined-RDT protocol** instead of UDP sockets. As you know the **Go-Back-N (GBN)** and **Selective Repeat (SR)** are the two protocols that are enhanced versions of the simple Stop-and-Wait protocol, your objective will be to write transport layer programs that implement GBN and SR protocols to upload the file reliably over unreliable medium. Please *note* that *the client must be able to transfer any kind of files (e.g., text, pdf, or media file) to the server*.

Disclaimer: This is a group assignment. Each group should have a max of two members. Therefore, it is up to you to find your teammate, but you need to divide the tasks equally among yourselves. You need to reach out to me quickly if you cannot find a partner or would like to do this assignment alone. There are no extra points given if you wish to do the assignment alone.

Task Details:

In the previous assignment (2a), you learned how to create a TCP socket and send data blocks (of 1000 bytes) to the client. However, in this assignment, you will be developing your own pipelined reliable transport protocol using GBN and SR that will deliver the packets reliably. Follow the below architecture to implement your programs.



Task-1 (File Transfer from Server to Client using GBN and SR)

Specifically, you should modify the server program to do the following,

1. When started, the server should ask for a port number and the protocol to use (GBN/SR). Then the server will wait for client's message starting with sequence# 0.
2. The first packet that it should receive is the command <RETR file_name>. With this, the server can prepare to send the asked file as per GBN/SR sender's specification.
3. Here, the server is going to play the role of sender as per our GBN and SR protocol's point-of-view. So, you will write programs/functions to ensure server behaves like a

GBN or SR Sender (based on the protocol chosen) that can handle sending packets in pipeline (with Window Size = N, you should try with N=4, 8, 12, 16, 24) and ensure sliding of window as acknowledgements are received. In addition, you need to make sure the segment size is 1024 bytes at maximum.

4. When the file's data is completely transferred, you can use your own END-OF-FILE signal to identify this situation.

Note: You would need to leverage **concurrent programming** concepts like multi-threading, locks/semaphores to build your GBN/SR sender functions.

Next, you need to modify the client program to do the following:

1. The client should ask the server's IP, port, and protocol to use as transport layer.
2. As per the protocol chosen, you need to include the GBN and SR receiver function in your client program.
3. As the first packet, the client should send the file name to retrieve from the server with a command "RETR *file_name*".
4. As the client program receives the packets, it needs to extract the content of correctly ordered packets and append to a new file (named *file_name*). It must continue this process until the whole content of the file is received.

Task-2 (Measurements)

As you have implemented the GBN and SR sender, you would need to count the following attributes to understand how this protocol is performing (conduct separately for transfer of both files). You need to send a large file for the sake of better visualization with N=4, 8, 12, 16, 24. For each protocol (GBN and SR), you need to measure following attributes.

i. Total number of packets sent (Both freshly transmitted and retransmissions)
ii. Number of retransmitted packets (only when time expires, packet gets retransmitted)
iii. Number of timeout events occurred
iv. Time taken to complete the file transfer

Then, use a spreadsheet or python's Matplotlib module to plot the following comparison figures

a. Window size (N) vs. time taken to complete file transfer
b. Window size (N) vs. # of retransmissions
c. Window size (N) vs. # of timeout

Sample Helper Module (Must be used)

No special python modules related to file transfer is allowed to be used, except the **sample helper modules given to you**.

You are given with 3-auxiliary helper modules: ***timer*, *packet*, *udt***. You must examine the helper modules to get acquainted on how and when to use them.

- Timer module: Provides *start(), stop(), timeout(), running()* functions
- Packet module: Provides *make(seqNo, data), extract(packet), make_empty()*
- udt module: Provides *send(packet, sock, addr), rcv(sock)* to send/recv from unreliable channel

Environment to Execute the Programs

- Install a UBUNTU 18+ VM on your VirtualBox and then you would have to install the CORE emulator on that machine. Please follow the instructions from https://github.com/coreemu/core.
- You can create a tree topology on your own as we have seen in the class and define your client node as well as server node.
- In the Ubuntu home folder, you can create two subfolders named "Server" and "Client", which are the home folders for your server and client programs respectively.
- In the server terminal, you would need to navigate to Server subfolder and execute your program using "python3 rftServer2.py". Similarly, in the client terminal you have to navigate to the Client subfolder to run the client program using "python3 rftClient2.py".

If for some reason, you cannot install CORE, you can simply use the same machine (localhost) with two separate terminals for executing your client and server programs. Please contact me ASAP, if you are confused on how to run your programs.

After the client receives the file, it terminates. Then, you should test the correctness of the files by using the DIFF command - **diff -s recvdFile sentFile**

Submission:

As outcomes of this assignment, you would need to submit the following:

1. Python programs/modules that could produce required outputs with proper code documentation. Lack of code documentation will cost 5% of the grade.
2. Report containing your strategy to solve the problem, evidence, and multiple execution samples (i.e., sending different kinds of files), instructions for running the submitted programs, and references used. Comprehensive report with no grammatical error is expected and it carries 10% of this assignment's total grade.

Group Activity:

- The students must work in group of 2. If you decide to work alone, you will have to complete both parts, there is no bonus for working alone.
- Each student's contribution should also be reported with a comparative chart as shown below and should be reported through the report.

Task lists	Student-1's contribution	Student-2's contribution
…	…	…

References:

[1] Socket programming in python: https://realpython.com/python-sockets/
[2] Python multi-threading: https://realpython.com/intro-to-python-threading/
[3] GBN and SR Animation: https://www2.tkn.tu-berlin.de/teaching/rn/animations/gbn_sr/
[4] GBN and SR Tutorial: https://www.javatpoint.com/sliding-window-protocol