Computer Networks **Instructor: Dr. Deepak Tosh**
Spring 2023 **Due Date:** 04/07/2023

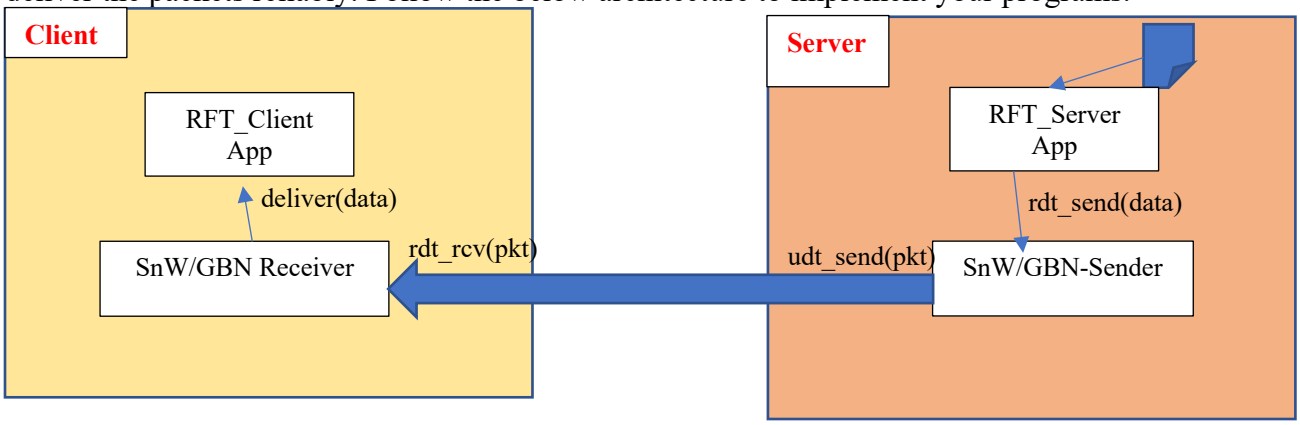
Assignment # 2b: Reliable File Transfer

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 (SnW) protocol, your objective will be to write transport layer programs that implement SnW and GBN 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. If one of your teammates is taking graduate version CS5313, your team then need to complete all the tasks. 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 Stop-and-Wait (SnW) and Go-Back-N (GBN) protocols that will deliver the packets reliably. Follow the below architecture to implement your programs.



Task – UG1 (File Transfer from Server to Client using Stop-and-Wait Protocol)

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

Experimental Setup Hint: You can think that the client is seeking to receive a file from the server, which means, the client will implement the receiver's logic and server will implement the sender's logic. In this case, the receiver program must be started first and then the sender program.

1. When started, the server should ask for a port number and the protocol to use (such as SnW or GBN). Then the server will immediately start sending a default file (it could be a text, pdf, docx, mp3, or mov file). You should choose different type of files to be sent in different execution runs and show evidence of file transfer.
2. Given the server is going to play the role of sender as per our SnW protocol's point-of-view, you will write programs/functions to ensure that server behaves like a SnW Sender (based on the protocol details discussed in class). In addition, you need to make sure the maximum segment size is 1000 bytes at maximum.
3. As you may remember, the sequence numbers used for SnW protocol are either 0 or 1, so make sure to use this while creating segments.
4. Similarly, modify the client program to run the SnW-Receiver's logic, where it initially waits for a segment with sequence number 0. You can make some assumptions on what type of file is being transferred and accordingly create the file with appropriate extension.
5. After receiving the file, make sure to use **diff** command to check the similarity or dissimilarity between the sender and receiver files.

Note-1: It is not required to include checksum in the segments. But you are welcome to implement this functionality**.**
Note-2: When the file's data is completely transferred, you can use your own END-OF-FILE signal to identify this situation.

Task – UG2 (Measurements for Stop-and-Wait Protocol)

As you have implemented the SnW sender, you would need to add few counters in your program to measure the SnW's performance.

For reporting, you need to measure following attributes.

i. Total number of transmitted packets.
ii. Number of retransmitted packets (only when time expires, packet gets retransmitted).
iii. Time taken to complete the file transfer.

Below tasks are mandatory for 5313 students/groups.

Task-G1 (File Transfer from Server to Client using GBN)

Now, you can modify the above program to add GBN's sender functionalities.

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

1. When started, the server should ask for a port number, the protocol to use (such as SnW or GBN), and window size (N). Then the server will immediately start sending a default file (it could be a text, pdf, docx, mp3, or mov file). You should choose different type of files to be sent in different execution runs and show evidence of file transfer.
2. Here, the server is going to play the role of sender as per our GBN protocol's point-of-view. So, you will write programs/functions to ensure server behaves like a GBN Sender that can handle sending packets in pipeline (with Window Size = N, you should try with N=4, 8, 12, 16) and ensure sliding of window as acknowledgements are received. In addition, you need to make sure the segment size is 1000 bytes at maximum.

3. When the file's data is completely transferred, you can use your own END-OF-FILE signal to identify this situation.
4. You need to carefully chose what sequence number range you would like to use and repeat using them after the segment uses the maximum value.

Note: You can leverage **concurrent programming** concepts like multi-threading, locks/semaphores to build your GBN sender functions. Or you can also leverage **select** function to handle socket read/write events asynchronously.

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

1. The client should ask the server's IP, port, protocol, and file_name to use as as part of reliable file transfer protocol.
2. As per the protocol chosen, you need to include the GBN receiver function in your client program.
3. 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-G2 (Measurements)

As you have implemented the GBN sender, you would need to count the following attributes to understand how this protocol is performing (conduct separately for transfer of different 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.

iv. Total number of packets sent (Both freshly transmitted and retransmissions)
v. Number of retransmitted packets (only when time expires, packet gets retransmitted)
vi. 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

- Use the same environment that was used in Programming Assignment-2a.

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 result 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 (pseudocodes can be great), 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. If you worked as a group, complete the below activity table. Also, provide evidence that the sent file and received files are exactly same (using **diff** command).

Group Activity:

- If you decide to work alone, you will have to complete all the required tasks, 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/