CS4133/5133: Data Networks

Fall 2017

100 pts

Project-4

Sliding Window Protocol Using UDP

Due Date: Nov 19, 2017

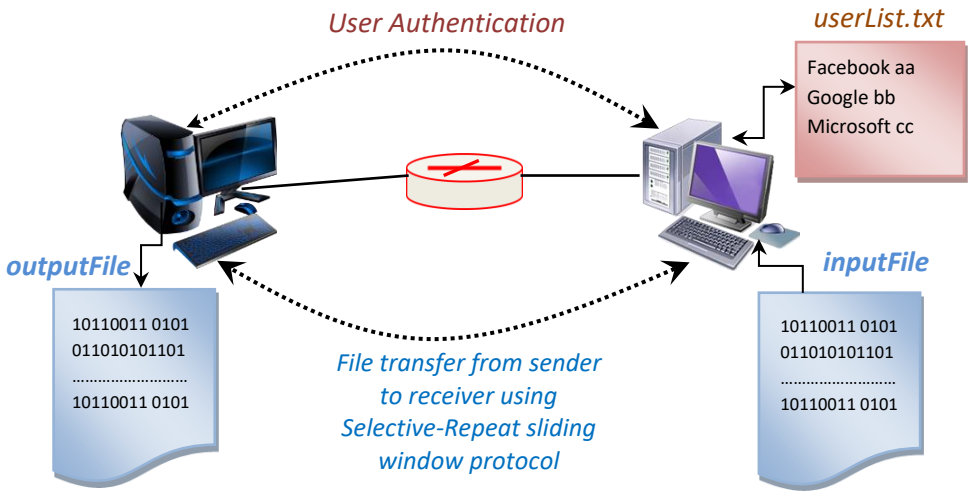
1. Objective:

The objective of this programming project 4 is to study and implement the Selective Repeat Sliding Window Protocol which is a Data Link Layer Protocol for reliable data transfer between network nodes. After completing this project, students will have an intermediate understanding of the steps required to control the flow of packets and adding reliability features in UDP applications for developing a networking application.

2. Project Specification:

2.1. Overall System Behavior:

You are required to implement a reliable simple file transfer protocol which uses the Selective repeat sliding window protocol for data and flow control. For detail about the Selective repeat sliding window protocol, please refer to your text book. Selective repeat is the basic mechanisms used in reliable window-based process to process communication protocol. The Selective repeat window protocol contains a *sender window.* There is also a *receiver window* that allows the receiver to buffer packets received in any order. The essence of the selective repeat protocols is that at any instant of time, the sender maintains a set of sequence numbers corresponding to frames it is permitted to send. These frames are said to fall within the sending window. Similarly, the receiver also maintains a receiving window corresponding to the set of frames it is permitted to accept. The receiver can explicitly request retransmission of a "missing" packet from the receiver by sending a *negative acknowledgement*. The following figure shows high-level overall system requirements.



2.3. Sender Behavior [Server]:

- At startup, the sender takes two arguments that specify the port number that it is listening to and the window size. You have to use (*5000+last 4 digits* of your student-id number) to avoid requesting same port by multiple students. The second parameter, window size of the sender, is set to be equal to 1.
- After successful startup, the sender program will wait for a UDP client interaction for **user authentication**. Server has *userList.txt* file which contains user name and password separated by a blank space. If authentication is successful, then sender will wait for a file transfer using UDP.
- After receiving a file name from the client, sender process will verify the existence and read permission of the file. If the file doesn't exist or permission denied, then the sender will transmit appropriate warning message back to client and wait for a new valid file name.
- If the file exists and read permission granted, then sender will send the window size (sender and receiver has same window size) and start transmitting data using UDP packet to client. You may use **128/256 bytes of payload** per packet.
- Your sender program should handle this packet transfer using Selective-Repeat Sliding Window Protocol discussed earlier.
- The protocol identifies the DATA segments by using **sequence numbers**. The sequence number of the first segment must be 0. It is incremented by one for each new segment. The receiver must acknowledge the delivered segments by sending an ACK segment.
- Along with each data packet, you need to deploy a **timer** to keep track of delayed or lost ACK/NACK packets from the receiver.
- As the sender program receives ACK from the receiver, it needs to randomly **drop 10%** of the ACK to simulate ACK lost from the receiver (generate random numbers to handle this requirement). You need to retransmit based on the Selective Repeat protocol before sliding the sender's window. The flow of data is unidirectional, meaning that the sender only sends DATA segments and the receiver only sends ACK segments. [Optional]
- Finally, when the data transmission is complete, your sender program will go back and wait for another UDP client communication for next user authentication and file transfer.

2.3.1. Sender Output:

- You need to print *SEQ number* and *type of operation* (transmission/retransmission) message after sending the data packet so that we can see your processes are working correctly (or not!).
- After successful transmission/retransmission of packets, when the file transfer is complete, your sender program should print the following **statistical report** about the whole transmission.
  - Receiver IP and Port Number
  - Name and Size of the file that was transferred
  - File Creation Date & Time
  - Number of Data Packets Transmitted

- Number of Packets Re-transmitted
- Number of Acknowledgement Received
- Number of Negative Acknowledgement Received with Sequence Number

2.4. Receiver Behavior [Client]:

- Your client program takes four arguments from user during startup that specifies the IP address and port number of the sender, input file name that receiver wants to copy from the sender and the output file name which will be used for creating the new file at receiver side.
- After a successful startup, your receiver program will print a welcome message and ask the user to input user name and password for login into the sender program (server).
- These data will be sent to the receiver program using UDP for the user authentication. Then, your sender, after receiving the user name and password from the receiver, will verify the received pair of username and password against the pairs in userList.txt file. If the result is positive, the server will send a success message to the receiver along with the window size (sender and receiver has same window size). If the result is negative, the server will send an error message to the sender and wait for the new pair.
- After successful authentication, your receiver program will ask the user whether to start the file transfer or not. With a positive feedback, the input file name will be sent to the sender program to verify the existence and access permission.
- If file doesn't exist or permission denied, the sender program will inform the receiver. You need to print the message and prompt the user for a new valid file name. If the file exists and read permission granted, then receiver will create an output file and start copying the incoming data into the file. *Note, UDP packets may come unordered. You need to keep track of it using the sequence number.*
- Your receiver program should handle this ACK/NACK packet transfer using Selective-Repeat Sliding Window Protocol discussed in the class.
- As the receiver program receives data packets from the sender, you need to randomly **drop 10%** of the data packets to simulate the data lost (generate random numbers to handle this requirement). You need to send NACK packet based on the Selective Repeat protocol before sliding receiver window for new set of packet(s). [This part is optional]
- When the file is completely received, your receiver will go back and ask the user whether s/he wants to send any more file or not.
- Finally, you need to verify whether the target file and the new file contents are exactly same or not. You may use *UNIX diff* command or any other appropriate command to verify. This will ensure file transfer without data loss and order. If it's an audio/video or picture file, then you may play the file or preview the picture for verifying the correctness of the new file.

2.4.1. Receiver Output:

- You need to print informative message after receiving a data packet and sending an ACK/NACK packets so that we can see your processes are working correctly (or not!).

- After successful file receive, your receiver program should print the following **statistical report** about the whole transmission.
  - Name and Size of the file that was received
  - File Creation Date & Time
  - Number of Data Packets received
  - Number of Acknowledgement Sent
  - Number of Negative Acknowledgement Sent

2.5. Input Format:

Sender and receiver must be command-line tools that allow transmitting one binary file and supporting the following parameters:

sender <port_number> <window_size>

receiver <sender_ip> <sender_port> <input_file> <output_file>

3. Programming Notes:

You are required to work on this project alone. You have to use UDP sockets for implementing both sender and receiver programs. You should program each process to print an informative statement whenever it takes an action (e.g., sends or receives a message, detects termination of input, etc), so that you can see that your processes are working correctly (or not!). One should be able to determine from this output if your processes are working correctly. You should hand in screen shots (or file content, if your process is writing to a file) of these informative messages. Note that we will be ***using our gpel machines (Unix platform)*** in the lab to run and verify your codes.

Make sure, your Sender (server) allows the user to specify the listening port number and window size during startup. You have to close every socket that you use in your program.

Many of you will be running the sender and receiver on the same UNIX machine, this is fine. Any clarifications and revisions to the project will be posted on the course web page on Canvas (canvas.ou.edu). Students are encouraged to start this project early and use the discussion list on Canvas for any general question so that everyone can benefit from the replies. Please do not use the list to flame others, and under no circumstances to post solutions for others.

4. File names:

File names for this project are as follows:

Client: *lastNameSelectiveRepeatReceiver.c*

Server: *lastNameSelectiveRepeatSender.c*

5. Points Distribution:

Bits and pieces	Points
Client Program	40
Server Program	40
Program Style (Coding style, comments etc.)	10
Documentation	10

6. Submission Instructions:

This project requires the submission of a *soft copy* and a *hard copy. Plagiarism* will not be tolerated under any circumstances. Participating students will be penalized depending on the degree of plagiarism.

Soft Copy (Due Nov 19, 2017, 11:59 pm)

The soft copy should consist of:

- source code of the Client program,
- source code of the Server program,
- any header file(s), and
- detailed documentation (submit as a **hard copy** also) should consist of:
  - discussion of your problem-solving approach,
  - detailed analysis of your implemented codes,
  - screen shots of outputs

7. Late Penalty:

Submit your project on or before the due date to avoid any late penalty. **A late penalty of *15% per day* will be imposed after the *due date*.** After one week from the due date, you will not be allowed to submit the project.

Good Luck!!