COM SCI 118 Computer Network Fundamentals

Project 2: Window-based Reliable Data Transfer over UDP
Fall 2014

1 Goal

The purpose of this project is to use UDP Socket and C/C++ programming language to implement a reliable data transfer protocol.

2 Instructions

1. In this project, you will implement a simple window-based protocol built on top of either Selective Repeat protocol or Go-Back-N protocol described in the textbook. It's up to you which protocol you will work on. Stop-wait and Stop-forward protocols are NOT allowed in this project. You must write one program implementing the sender side, and another program implementing the receiver side. Only C and/or C++ are allowed in this project. No 3rd-party libraries are allowed.

2. The two programs should communicate the User Datagram Protocol (UDP), which does not guarantee data delivery.

3. The receiver program will also act as a client application, requesting a file from the sender side.

4. The receiver program will take the hostname and port number of the sender, and the name of a file it wants to retrieve from the sender as a command line arguments. For example:
   to run the sender program: $shell > sender < portnumber >$
   to run the receiver program: $shell > receiver < sender\_hostname >< sender\_portnumber >< filename >$

5. The receiver will first send a message to the sender which includes the name of the file requested. If the file exists, the sender will divide the entire file into multiple packets (the maximum packet size is 1K bytes), and then add some header information to each packet before sending them to the receiver. It is up to you what information you want to include in the header (e.g. Source, Destination port etc...), but you will at least need a sequence number field. You are free to define what kind of messages you will require, and the format of the messages. You can create a large file being requested on your own, but make sure the file is transmitted in multiple packets.

6. Note that your programs will act as both a network application (file transfer program) as well as a reliable transport layer protocol built over the unreliable UDP transport layer.

7. Congestion control is not mandatory in this project. But if you implement it, we would consider extra credits for this project. For backward compatibility to the normal code, your congestion control algorithm should be implemented as an optional feature. That is,
   to run the sender program: $shell > sender < portnumber >< enable\_congestion\_control >$
   to run the receiver program: $shell > receiver < sender\_hostname >< sender\_portnumber >< filename ><$ $enable_congestion_control >$

8. You should print messages to the screen when the server or the client is sending or receiving packets. Your message should include information such as whether it is a DATA packet or an ACK, the sequence number, whether it is corrupted or lost etc. If you have implemented the congestion control, you should also show the *cwnd* and *ssthresh*. Such messages will be helpful for you to debug the programs, and we can use them to examine the correctness of your programs.

If *enable_congestion_control* = 1, the congestion control algorithms should be launched, otherwise your program should work as the normal case.

3 Emulate Packet Loss and Corruption

Although using UDP does not ensure reliability of data transfer, the actual rate of packet loss or corruption in LAN may be too low to test your program. Therefore you should simulate packet loss and corruption in the following manner:

- Packet loss: With probability P_l ignore arriving packets (pretend not receiving the arriving packets).
- Packet corruption: With probability P_C mark an arriving packet as being corrupted (pretend the arriving packets are corrupted).
- P_l and P_C range between 0 and an appropriate value (say, 0.40). Note both data packets traveling from sender to receiver and acknowledgement packets traveling from receiver to sender may be lost or corrupted.

4 Due Time and Demo

1. You are required to demo your program on Monday(12/8) and Wednesday(12/10).
2. Submit an electronic copy of the project on CCLE on 11:59:59pm Friday (12/5).

4.1 Demo

In the demo, we provide you the provided VM machine. You then use make to compile your programs, run your programs to deliver a test file from the sender side to the receiver side. It is required by your program to print out the operations, which explain the delivery process on the screen. We may ask you to use different values for P_l and P_C to test your programs. We will ask you to compare the received test files with the original one using the tool ”diff”.

Demo Procedure:

1. Sign up for Demo : TA will distribute the signup sheet in the discussion section of the 6th and 7th week.
2. Each team has up to 10 minutes to demo. No time extension would be given. So please make sure your demo is fast and easy to understand.
3. You are allowed to change your code during the demo, but no extra demo time would be given.
4. In Demo : You need to prepare 3 slides to present. 1 slide for design and implementation, 1 slide for experiences you gain, 1 slide for lesson learn from project and suggestion to project.
   - TA will ask you to demo the function step by step
   - TA will also ask you questions during demo, you need to answer the question clearly. All question will related to your project implementation.
5. If you have implemented the congestion control, you should report which functions you have implemented (slow start, congestion avoidance, fast retransmit/recovery, etc.). Your demo should clearly show how the *cwnd* and *ssthresh* evolves during the data transfer.

5 Project Submission

1. Put all your files into a directory, must called *project2_UID.tar*. UID is the student id of one of the students of the group.

2. Submit the file *project2_UID.tar* via SEAS online submission in course webpage.

3. The *project2_UID.tar* should contain the following files

   - Source codes (can be multiple files)
   - A report file (.doc or .pdf) no more than 3 pages. The report will contain:
     - Student names and Student IDs at the very beginning (2 students per-group).
     - Implementation description (header format, messages, timeouts, window-based protocol etc).
     - Difficulties that you faced and how you solved them.
   - Makefile

4. The TAs will only type “make” to compile your code, make sure your Makefile works in the provided VM machine.

5. Each group just needs to submit one set of files.