CS F303 Computer Networks: 2020-21 semester II

Assignment 2
60 Marks

Do this assignment in groups of 4-5-6 (not less than 4, not more than 6, all groups will be treated at par without taking into account the number of members). Only one group member is required to submit for each phase of the assignment.

You are developing an application which can't tolerate the growth and shrinking of the TCP congestion window, but requires reliable delivery. Therefore, you are required to use UDP as the transport protocol, but build reliability in the application layer.

Phase I (Deadline: 19th March, 2021 23:59 hours) [15 marks]
This phase is to be done by 2 groups together (3 groups if there is one group left out).

1. 2 (or 3 groups) are required to submit a Design Document for this phase. You can structure it on the lines of an RFC. The Document should outline the protocol that the 2 (or 3) groups design together
2. Design your own application layer middle-ware protocol which uses UDP sockets for communications
3. You should choose the client-server paradigm
4. Your protocol should adhere to the **Selective Repeat paradigm**
5. Your protocol should be able to take care of sequence numbers, acknowledgements, negative acknowledgements, retransmissions, and what ever else is required to run your application smoothly.
6. The specification document should contain the assumptions about the application, the network, the strategies to tackle different situations in the network, etc.
7. The assignment evaluation will include degraded network conditions using the *netem* tool. See the Evaluation section below.
8. What to submit: One PDF submission by any member of all the groups is required. Group wise names and ID number of all members should be present on the top of the document. Maximum length of this document is 2-3 pages, 11 point font for non header text.

Phase II (Deadline: 11th April, 2021 23:59 hours). [45 marks]

1. You can start Phase II at any time, without waiting for the Phase I deadline to be over
2. **Implementation A:** In this phase the groups need to independently implement the protocol they have designed in Phase I. That is, if Group A and Group B had together designed the protocol, in this phase they should independently implement the protocol. Independent implementation is compulsory and will be verified for evaluation.
3. You can use any programming language you are comfortable with, however the 2 (or 3) groups who have designed the protocol together should use different languages. That is, if Group A chooses to use Java then Group B can choose to use Python (not Java). The protocol implementation code needs to be submitted by one member of each group.
4. **Implementation B:** Write a network file transfer application (client-server) which uses the protocol implementation you designed in the previous step. The application and the protocol codes need to be decoupled compulsorily. Decoupling can be done by importing the protocol implementation as a library, class, header file, etc. Use the same programming language used in implementing the protocol.
5. The applications and protocols designed by the 2 (or 3 groups) in Phase I must be inter-compatible. That is, Group A's client should work with Group B's server implementation and vice versa, independent of the language used to implement them. This will be checked during evaluation. This is the essence of designing a network protocol: that implementations are independent and inter-compatible.

6. The protocol should ensure that application works under different network conditions. See the Evaluation section below.
7. What to submit: (A single zip per group)
   a) Each group needs to submit their protocol and application implementation files. Only one group member needs to submit. Include the names and ID numbers of all group members in all files as comments.
   b) Each group should include a Readme file with instructions to compile and run your protocol and application. Also include in the Readme, as a separate section, any changes in the protocol that you had to make during the implementation and testing in Part II.
   c) Each group should submit a pdf file with plots of the performance of your protocol under different network conditions. Plots to include are throughput vs packet loss, delay, packet reordering, packet corruption, etc. (use the netem tool to emulate different network conditions on your localhost, see the Evaluation section below for *netem*). If you have written scripts to generate the plots, submit them as well.

Evaluation: Submit a single zip file with all the above asks for each phase. Only one group member is required to submit. Your submissions should run on Ubuntu 18.04 or 20.04. Your application will be tested under various network conditions using the *netem* tool (https://wiki.linuxfoundation.org/networking/netem). e.g. 50% packet loss, 90% packet loss, varying delays, jitter, packet corruption, reordering, rate, etc. Your protocol should be reliable under these conditions. We will also use different file sizes and different file formats. You will be required to demo the assignment to the TAs or instructors or both. The 2 (or 3) groups will demo their assignment together. The modalities of the demo will be communicated later.

Collaboration within a group: Use Google Meet, Google Docs, Gitlab (private), etc. to collaborate.

Collaboration outside the 2 (or 3) groups working together: The class is encouraged to discuss and collaborate, but cheating in any form is strongly discouraged. Use piazza for discussions so that all discussions are in the open. **Include a file with the names of people you have discussed with to maintain transparency.** There is almost no scope of overlap between two groups. Your submission will be passed through code similarity and document similarity checkers. Any form of cheating will be dealt with along declared policies. The bottom of this page has a good reference on cheating vs. collaboration: http://www.cse.iitd.ac.in/~mausam/courses/col772/spring2019/

Start early, the deadline is a hard deadline as the institute has a hard deadline for pre-comprehensive evaluations. Divide the days into protocol design, protocol implementation and application implementation based on how much effort each of these components requires. Hint: a clear protocol design, covering all corners is easier to implement.