CSC 257/457: Computer Networks — Fall 2001

Project Two: Reliable Datagram Protocol

Assigned: Tuesday, October 23rd, 2001
Project design due: by in class, Tuesday, October 30th, 2001
Due: Monday, November 12th, 2001, midnight.

In this project, you will design, implement, and evaluate a reliable datagram protocol (RDP), given the following:

- a description of the communications service that your protocol should provide to its clients, and
- a description of the communications service that is provided to your protocol by the underlying communications link.

Your task is to

- design a protocol that correctly and efficiently implements the required communications service,
- specify the protocol in sufficient detail to convince the reader that it correctly implements the specified communications service,
- implement and debug the protocol, and
- evaluate the protocol's performance.

1 Reliable Datagram Service

This section describes the communications service your protocol should provide. The interface should be similar to that provided by UDP, except for requiring connection establishment and termination (as in TCP). A successfully closed connection should imply that all messages were delivered *exactly once*. None of the operations exported by either the protocol or the client may block indefinitely. The probability that a message is delivered to the client with altered contents must be very low. Messages must be delivered in the same order they are sent. The maximal message size supported must be no less than 32,500 bytes.

In order to get you started and to ensure a common interoperable service interface, we have provided a skeleton `rdp_lib.c` in `/u/csc257/projects/rdp/skel`. `/u/csc257/projects/rdp` is the project directory. Other useful tests and routines may also be found in this directory. There is also a file outlining reporting requirements.

2 Underlying Communications Service

Your protocol should run on top of UDP, an unreliable, best-effort datagram communication service. UDP provides you with the necessary demultiplexing service for process-to-process communication, so you do not have to replicate this in your transport layer. However, UDP may not deliver some messages, may deliver messages out of order, and may deliver duplicates of a message. The maximal frame size supported by UDP is 32768 bytes. Your protocol should never inject more than 100 KBytes of unacknowledged user data into the network.

3 Protocol Design

You have a large degree of freedom in designing your protocol. Some of the alternatives you should consider include

- packet size
- sliding window vs. concurrent logical channels
- selective repeat vs. go back n
- ACK vs. ACK/NAK
- piggybacking or not
- RTT estimation

Your protocol should be well-documented, both in terms of its functionality (service interface), and in terms of its peer interface. Be sure to weigh the alternatives whenever you are facing a decision point. Make a conscious decision, write down the reason for making the decision, and the assumptions that this decision is based on. Your design will be evaluated in part based on its coherence.

To provide you with feedback on your design, you are required to submit a design document by Tuesday, October 30th. Your document should present a coherent design, a rationale for each of your design choices (why did you decide to do X, and on what assumptions was this decision based), and a specification of your protocol. Specify the exported functions and write up a walk through of what happens when the user-level program does a send of a small amount of data and what happens when the user-level program does a send of a large amount of data. The design portion of your grade will be based in part on this document.

4 Protocol Evaluation

Configure your protocol on top of UDP as a library that can be linked in with any of your test programs/applications. To measure round-trip latency, measure how long it takes to send 10,000 RDP messages of size 1-byte, 200-bytes, 400-bytes, ..., and 1400-bytes from the client to the server, and back to the client. To measure throughput, use RDP to send 10,000 messages of size 1 KByte, 2 KBytes, 4KBytes, and 8 KBytes from the client to the server. Explain your measured results. Compare the results you obtained to the performance of TCP with a similar experiment.

To avoid flooding the network, test your protocol on a single machine before moving to multiple machines. Also, avoid performance measurements during peak lab usage times.

Configure your protocol library with a compile-time flag to drop every 10th message (there are some simple tests in the project directory to help you with this). Repeat the experiments on throughput and latency to determine both the correctness and performance of your protocol. Also, configure your protocol to introduce additional delays on receiving a message in order to simulate a different RTT. Determine ensuing performance.

5 Grading

This project is a *group* project (both for those taking the course as CSC257 and for those taking the course as CSC457) — groups can consist of one or two people. You are on your own to form your groups. Groups cannot be formed or changed after the design document is due. Part of your external documentation should be a discussion of who did what part of the work.

You should turn in

- your revised and final design document,
- the source code for your protocol,
- a brief description of your experiments, and a summary and explanation of your results.

Be sure to include appropriate Makefiles and README files.

Your work will be graded based on the following criteria:

- correctness, elegance, and coherence of your design
- performance (bandwidth, latency, host processing overhead)

After the due date, you should schedule 15-minute sessions with Chunqiang Tang to demonstrate your software, and explain how you collected your results.