Lab 2: Reliable Transport over UDP

14–740 Fundamental of Telecommunication Systems

Released: March 21, 2012, 11:59pm EDT
Due: April 21, 2012, 11:59pm EDT

1 Introduction

In this project, you will be exploring various functions that are offered by a reliable transport layer protocol. After successfully completing this lab, you would have built a reliable transport protocol (like TCP). Lets call this protocol Trusted Transport Protocol (TTP). TTP will have many of the properties that TCP has, like acknowledgements, re-transmission, segmentation and re-assembly, sliding window mechanisms, etc. You will use UDP as the underlying mode of communication between nodes, on top of which you will add some properties that ensure reliability.

2 Part 1 – Getting Started

Figure 1 shows a diagrammatic representation of the layers that you should be familiar with before starting the lab.

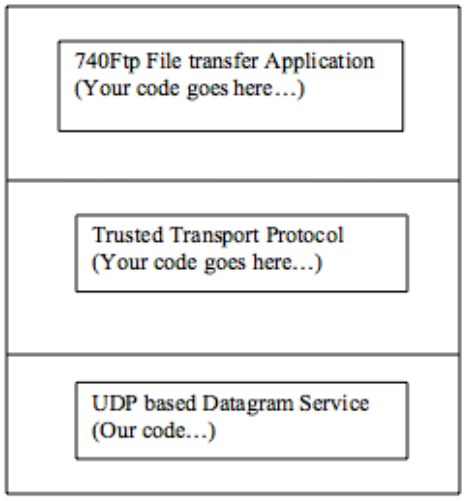

Figure 1: Software Stack

3 Programming Environment and Logistics

We expect you to write code for the project in either C or Java. Each language has its own pros and cons. You are advised to choose the language you are most comfortable with. Make sure your code runs on Linux machines. You will be working in groups of 2 so the very first step in doing the lab is to "Go find a partner!".

4 Support Code

The support code is available in both C and Java. The Java version can be directly imported into Eclipse as a project. Download appropriate tar-files from blackboard. The support code contains code for Datagram Service (details below) and a sample client server application that demonstrates datagram service for communication. *Note: the client-server application is not over a reliable channel*.

5 Datagram Service

This layer has the following properties

1. We use UDP as the transfer protocol to send/receive messages between nodes. This means that we use only Datagram sockets for communication at this layer.
2. This layer exports 2 facilities to upper level protocols
   - `send_datagram()` to send a single datagram
   - `receive_datagram()` to receive a single datagram
3. The format of the datagram this layer uses is shown in Figure 2. *All the fields of the datagram are explained in support code. Except payload, everything else is of fixed size.*

Source IP	Destination IP	Source port	Destination Port	Size of payload	Checksum	Payload (Max 1460 bytes)

Figure 2: Datagram Format

4. The Datagram Service needs to be initialized using `init_datagram_service()`. This essentially sets up a socket for sending/receiving datagrams.
5. The Datagram Service does not calculate or validate checksum. You will need to do this in your implementation of TTP. As far as checksums are concerned, you should follow the same procedure/algorithm for computing checksums as used by UDP/TCP but you can ignore the IP part of the header in computing the checksum.

Note: The exact functions names in C and Java implementations differ. Please refer support code.

6 Trusted Transport Protocol (TTP)

We give you complete freedom in design of the Trusted Transport Protocol i.e. you may choose your own payload structure (with required parameters such as sequence numbers), API exposed to upper layers, types of messages exchanged (e.g. SYN, ACK, DATA) etc.

Your TTP implementation should adhere to following requirements

1. It should use the Go-Back-N model for achieving reliability and in-order delivery.
2. Following parameters should be configurable via command line during node initialization
   (a) Sender/receiver window size
   (b) Retransmission timer interval
3. TTP should be robust against following types of error conditions
   (a) Duplicate datagrams
   (b) Delayed datagrams
   (c) Dropped datagrams
   (d) Checksum errors (only single bit flips in payload section of datagram)
4. It should use our Datagram Service for all communications over network.
5. The file transfer application may ask TTP to transfer data that does not fit in a single datagram. TTP should handle segmentation and reassembly of data.
6. TTP should at least expose four APIs to perform following functions
   (a) Start a new connection
   (b) Send data over a connection
   (c) Receive data
   (d) Close a connection
7. It should be able to handle multiple connections simultaneously.

7 File Transfer Application (740Ftp)

In order to demonstrate correct working of your TTP implementation, you will develop a simple file transfer application (called 740Ftp) that uses TTP. 740Ftp should follow a simple client-server model.

740Ftp-server: Maintains a set of files which are sent to clients upon request.

740Ftp-client: Contacts the server to obtain a file.

A typical file transfer session should involve following steps

- A user starts the 740Ftp-client with required filename and 740Ftp-server information
- Client initiates connection to the server using TTP.
- Client sends request for the file to the server over established connection
- Server searches for requested file locally, reads the file and transfers data back to client over the open connection.
- Client stores the data received in a local file and closes connection.

The 740Ftp application should ensure that the file is transferred without any errors. Note that TTP can identify only single bit errors and thus upper layers should ensure application level error detection. A simple way to achieve this would be asking the server to send MD5 hash of the entire file to client and verifying it on client side. The 740Ftp-server should run forever serving multiple clients. The 740Ftp-client may exit after a file is successfully transferred (or error is received). The server should be able to serve arbitrarily large files.
Please do not develop any GUIs. We expect 740Ftp to be a command line tool.

8 Testing your code

In order verify correct behavior of TTP and 740Ftp you will need to simulate various error conditions. You can use the Datagram Service code to simulate packet drops, delays, out-of-order deliveries etc. This is exactly why we gave you the Datagram Service code ☺. We get to control it and test your TTP and FTP740 implementations thoroughly by simulating error conditions! During evaluation, we will be replacing the Datagram Service code with modified version of the code that provides same API. So do not make any changes to the data structures and API in Datagram Service. We will provide you a set of test cases closer to the due date.

Extra Credit: You can submit your version of Datagram Service with tests cases. We will assign extra credit depending on how exhaustive the tests are.

9 Hand-in Instructions

1. Make sure you submit all relevant source files. Clean your project before submitting.
2. Prepare a write-up describing in detail the design of TTP and 740Ftp.
3. Make sure your code is well documented.
4. Submit your code and write-up in a tar file ($< userid1 > \_ < userid2 > .tar$) on blackboard

10 Grading Criterion

25%: Overall architecture and design
45%: Correctness of TTP implementation
20%: Correctness of 740Ftp implementation
10%: Documentation and coding style
10%: Extra credit!