CSC 573 – Internet Protocols
Project #2
Spring 2014

Project Objectives

In this project, you will implement the Go-back-N automatic repeat request (ARQ) scheme and carry out a number of experiments to evaluate its performance. In the process I expect that you will develop a good understanding of ARQ schemes and reliable data transfer protocols and build a number of fundamental skills related to writing transport layer services, including:

- encapsulating application data into transport layer segments by including transport headers,
- buffering and managing data received from, or to be delivered to, the application,
- managing the window size at the sender,
- computing checksums, and
- using the UDP socket interface.

Simple File Transfer Protocol (Simple-FTP)

The FTP protocol provides a sophisticated file transfer service, but uses TCP to ensure reliable data transmission. You will implement Simple-FTP that provides a simple service: transferring a file from one host to another. However, Simple-FTP will use UDP to send packets from one host to another, hence it has to implement a reliable data transfer service using the Go-back-N ARQ scheme. Using the unreliable UDP protocol allows us to implement a "transport layer" service such as reliable data transfer in user space.

Client-Server Architecture of Simple-FTP

To keep things simple so that you focus on the details of the Go-back-N protocol, you will implement Simple-FTP in a client-server architecture and omit the steps of opening up and terminating a connection. The Simple-FTP server will play the role of the *receiver* in the reliable data transfer, and the Simple-FTP client will play the role of the *sender*. All data transfer is from sender (client) to receiver (server) only; only ACK packets travel from receiver to sender.

The Simple-FTP Client (Sender)

The Simple-FTP client implements the sender in the reliable data transfer. When the client starts, it reads data from a file specified in the command line, and calls `rdt_send()` to transfer the data to the Simple-FTP server. For this project, we will assume that `rdt_send()` provides data from the file on a byte basis. The client also implements the sending side of the reliable Go-back-N protocol, receiving data from `rdt_send()`, buffering the data locally, and ensuring that the data is received correctly at the server. The client also reads the value of the maximum segment size (MSS) from the command line. The Go-back-N buffers the data it receives from `rdt_send()` until it has at least one MSS worth of bytes. At that time it forms a segment that includes a header and MSS bytes of data; as a result, all segments sent, except possibly for the very last one, will have exactly MSS bytes of data.
The client also reads the window size N from the command line, and implements the sending size of the Go-back-N protocol. Specifically, if less than N segments are outstanding (i.e., have not been ACKed), it transmits the newly formed segment to the server in a UDP packet. Otherwise, it buffers the segment and waits until the window has advanced to transmit it. Note that if N = 1, the protocol reduces to Stop-and-Wait.

The header of the segment contains three fields:

- a 32-bit sequence number,
- a 16-bit checksum of the data part, computed in the same way as the UDP checksum, and
- a 16-bit field that has the value `0101010101010101`, indicating that this is a data packet.

For this project, you may have the sequence numbers start at 0.
The client implements the full Go-back-N protocol as described in the book, including setting the timeout counter, processing ACK packets (discussed shortly), advancing the window, and retransmitting packets as necessary

The Simple-FTP Server (Receiver)
The server listens on the well-known port 7735. It implements the receive side of the Go-back-N protocol, as described in the book. Specifically, when it receives a data packet, it computes the checksum and checks whether it is in-sequence, and if so, it sends an ACK segment (using UDP) to the client; it then writes the received data into a file whose name is provided in the command line. If the packet received is out-of-sequence, or the checksum is incorrect, it does nothing.

The ACK segment consists of three fields and no data:

- the 32-bit sequence number that is being ACKed,
- a 16-bit field that is all zeroes, and
- a 16-bit field that has the value `1010101010101010`, indicating that this is an ACK packet.

Generating Errors
Despite the fact that UDP is unreliable, the Internet does not in general lose packets. Therefore, we need a systematic way of generating lost packet so as to test that the Go-back-N protocol works correctly (and to obtain performance measurements, as will be explained shortly).
To this end, you will implement a *probabilistic loss service* at the server (receiver). Specifically, the server will read the probability value p, $0 < p < 1$ from the command line, representing the probability that a packet is lost. Upon receiving a data packet, and before executing the Go-back-N protocol, the server will generate a random number r in $(0, 1)$. If $r \leq p$, then this received packet is discarded and no other action is taken; otherwise, the packet is accepted and processed according to the Go-back-N rules.

Command Line Arguments
The Simple-FTP server must be invoked as follows:

```
Simple_ftp_server port# file-name p
```

where `port#` is the port number to which the server is listening (for this project, this port number is always 7735), `file-name` is the name of the file where the data will be written, and p is the packet loss probability discussed above.

The Simple-FTP client must be invoked as follows:

```
Simple_ftp_server server-host-name server-port# file-name N MSS
```

where `server-host-name` is the host name where the server runs, `server-port#` is the port number of the server (i.e., 7735), `file-name` is the name of the file to be transferred, `N` is the window size, and `MSS` is the maximum segment size.

Output
The code you submit must print the following to the standard output:

- Simple-FTP server: whenever a packet with sequence number X is discarded by the probabilistic loss service, the server should print the following line:

```
Packet loss, sequence number = X
```

- Simple-FTP client: whenever a timeout occurs for a packet with sequence number Y , the client should print the following line:

```
Timeout, sequence number = Y
```

Task 1: Effect of Window Size N

In this and the next two tasks, you will carry out a number of experiments to evaluate the effect of the window size N, MSS, and packet loss probability p on the total delay for transferring a large file. To this end, you must select a file that is at least 1MB in size, and run the client and server on two different hosts separated by several router hops; for instance, run the client on your laptop/desktop connected at home and the server on an EOS machine on campus. Record the size of the file transferred and the round-trip time (RTT) between client and server (e.g., as reported by `traceroute`), and include these in your report. For this first task, set the MSS to 500 bytes and the loss probability $p = 0.05$. Run the Go-back-N protocol to transfer the file you selected, and vary the value of the window size N = 1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024.

For each value of N, transmit the file 5 times, time the data transfer (i.e., delay), and compute the average delay over the five transmissions. Plot the average delay against N and submit the plot with your report. Explain how the value of N affects the delay and the shape of the curve.

Task 2: Effect of MSS

In this experiment, let the window size N = 64 and the loss probability $p = 0.05$. Run the Go-back-N protocol to transfer the same file, and vary the MSS from 100 bytes to 1000 bytes in increments of 100 bytes. For each value of MSS, transmit the file 5 times, and compute the average delay over the five transmissions. Plot the average delay against the MSS value, and submit the plot with your report. Discuss the shape of the curve; are the results expected?

Task 3: Effect of Loss Probability p

For this task, set the MSS to 500 bytes and the window size N = 64. Run the Go-back-N protocol to transfer the same file, and vary the loss probability from $p = 0.01$ to $p = 0.10$ in increments of 0.01. For each value of p transmit the file 5 times, and compute the average delay over the five transfers. Plot the average delay against p, and submit the plot with your report. Discuss and explain the results and shape of the curve.

Grading

Simple-FTP server and client code (compiles and runs correctly)	70 Points
Task 1	10 Points
Task 2	10 Points
Task 3	10 Points
	100 Points

Extra Credit: 10 Points

For an additional 10 points you may implement the Selective Repeat ARQ protocol (in addition to the Go-back-N protocol), and carry out Tasks 1-3 for Selective Repeat. However, make sure that you complete the Go-back-N implementation first, before attempting this task.