CS 450 – Introduction to Networking – Spring 2014
Homework Assignment 3
File Transfer using Go Back N

Due: **Monday 7 April. Electronic copy due at 10:30 A.M., Optional paper copy may be handed in during class.** (Note: the paper copy must match the electronic copy exactly.)

Overall Assignment

For this assignment you are to extend your earlier client and server programs to implement the Go Back N pipelining protocol.

New System Calls for This Assignment: alarm(2), signal(2)

Server Operation and Details

- The syntax for running the server shall be:

  ```
  myServer [ port ] ...
  ```

  - port is the port to listen to for incoming UDP Packets. Default value is 54323.
  - You may support additional arguments to appear after the ones shown here, provided they are well documented in your README file.

- The goal of the server is to accept incoming UDP Packets and to assemble them into files, with the following additional details:

  - For this assignment Packet size is fixed, according to the provided header file. The last Packet of a file may not be completely full, but this is handled by the nbytes and nTotalBytes fields.

  - Sequence numbers refer to Packets, not bytes. **THE SEQUENCE NUMBER FOR THE FIRST PACKET OF A FILE SHALL BE 1.**

  - The transaction number (and UIN number) uniquely identify a particular file transfer. It is acceptable (and expected) that your server only deal with one transaction at a time, and that it drop all incoming Packets not associated with the current transaction.

  - The server shall send back acknowledgements to the client every time an incoming Packet arrives, with the acknowledgement number equal to the sequence number of **the last Packet that was received successfully.** If no Packets have been successfully received then the server should send back a Packet with an acknowledgement number of zero.

Client Operation and Details

- The syntax for running the client shall be:

```
myClient [ server ] [ port ] [ relay ] [ relay port ]
        [ client port ] [ garbleChance ]. . .
```

  - server is the name or IP address of the computer running the server program. Default is localhost, 127.0.0.1
  - port is the port on the server that is listening for TCP connection requests, default 54323.
  - relay is the remote relay to use, or "none". Default is "none".
  - relay port is the port the relay is using. Default is 54322.
  - client port is the port the client should use. Default 54324.
  - garbleChance is an integer from 0 to 100 for the percent chance packets get garbled by the relay. By default the other related parameters (e.g. dupeChance) should be the same as garbleChance, but you may want to use seperate values while debugging.
  - If any argument is omitted, then all following arguments must also be omitted.
  - You may support additional arguments to appear after the ones shown here, provided they are well documented in your README file.

- The client shall (ultimately) perform the following tasks:

  - Ask the user for the name of a file.
  - Memory map the file to an address using mmap, to save actually reading it.
  - Note the time, for timing purposes.
  - Transfer the file to the server using UDP and the Packet definition provided on the web site, using the Go Back N pipelining protocol. (Set the protocol field in the packets to 31. See header comments for details.)
  - When the entire file transfer has been acknowledged, report the time and overall transfer rate.
  - If the user has additional files to transfer, repeat as necessary, **using a fresh transaction number**. Note that if your client shuts down and starts up again, it should restart with a fresh transaction number, and not re-use any recently used transaction numbers.

Evolutionary Development

It is recommended that the program be developed in the following order, making sure each step is working before beginning development on the next step:

1. Develop initially with the client and server running on the same computer, or on two adjacent computers in the computer lab, with no relay involved.
2. When that is working, incorporate a relay, but leave all the garbling parameters set to zero.
3. Then start increasing the amount and types of garbling done by the relay, to see just how bad of a connection your program can tolerate. In the end all of the garbling parameters (chanceGarble, chanceDrop, chanceDelay, chanceDupe) should be set to the same value, but for debugging purposes you may want to adjust them one by one.
4. Work out any remaining bugs so that the client-server system efficiently provides accurate file transfer and data transfer rate reporting.

5. Run the program over a range of different distances using files of varying sizes, from a single byte to a file size that takes about a minute to transfer the file and get the acknowledgements back, using the provided relays as necessary.
6. Optional Enhancements – See below.

Data Header Format and Contents

A defined data header will be provided on the web site, and may be subject to slight changes as needed. As of this writing, version 7 of the header file defines the following fields, **<u>which must be in this order</u>**:

- int32_t version; // Set to 7. Later versions may have more fields
- int32_t UIN; // Packets with unrecognized UINs will be dropped
- int32_t transactionNumber; // To identify parts of one transaction.
- int32_t sequenceNumber;
- int32_t ackNumber; // Acknowledgement number
- uint32_t from_IP, to_IP; // Ultimate destination, not the relay
- uint32_t trueFromIP, trueToIP; // AWS may change public IP vs private IP
- uint32_t nbytes; // Number of bytes of data to follow in this packet
- uint32_t nTotalBytes; // Total number of bytes in the file
- char filename[32]; // the name of the file being transmitted.
-                     // i.e. the name where the server should store it.
- // Then the 16-bit ones - An even number of these.
- uint16_t from_Port, to_Port; // Ultimate source & destination, Not relay
- uint16_t checksum; // One's complement of sum of remaining bytes
- // Then some 8-bit numbers. Hopefully a multiple of 4 of these
- int8_t HW_number; // e.g. 1 for HW1, etc.
- int8_t packetType; // 1 = file transfer; 2 = acknowledgement; 3 = both
- int8_t relayCommand; // 1 = close connection?
- int8_t saveFile; // non-zero: Save to disk. Else discard.
- char ACCC[8]; // your ACCC user ID name, e.g. astudent42
- int8_t dropChance; // 0 to 100, as an integral percentage
- int8_t dupeChance; // 0 to 100, as an integral percentage
- int8_t garbleChance; // 0 to 100, as an integral percentage
- int8_t delayChance; // 0 to 100, as an integral percentage
- uint8_t windowSize; // 0 to 255. Set to 0 for Stop-And-Wait
- int8_t protocol; // * 10. E.g. 22 => 2.2, 30 => 3.0, etc.
-     // 30 for RDT 3.0 stop-and-wait, 31 for RDT 3.0 Go-Back-N,
-     // and 32 for RDT 3.0 Selective_Repeat
-
- // Students may change the following, so long as the total overall size
- // of the struct does not change.
- // I.e. you can add additional fields, but if you do, reduce the size of
- // the reserved array by an equal number of bytes.
-
- char unused[409]; // Unused padding Total header size should be 512

Remote Relays

In order to provide for long distance data transfers, a set of relays will be provided at known locations.

- Each relay will listen for new UDP Packets at port 54323.
- When sending data to the relay for ultimate delivery to a server:
  - The To_IP field in the CS450 defined header should be the IP address of the server, i.e. the ultimate destination, not the IP of the relay.
  - The address specified in the sendto( ) call should be the address of the relay, not the server.
  - (When not using a relay, the two addresses are both the same, that of the server.)
- For any Packet arriving correctly at the relay it will be forwarded to the to_address and port, subject to the actions of the garbling parameters.
- Any data arriving at the relay without a valid header, or without an approved UIN number, will be dropped.
- The garbling parameters are expressed as the percentage chance (in whole numbers 0 to 100) that a particular type of garbling will take place. In the event that a Packet is both duped and garbled, then each copy will be garbled (or not) independently of the other copy(ies).

Required Output

- All programs should print your name and CS account ID as a minimum when they first start.
- Beyond that, this program should perform the work described above.
- It may be appropriate to report some summary statistics when the program closes, such as the total amount of data transferred, average response rate, etc. More details on this may be provided in class and/or on the class web site.

Analysis

Once the program is up and working, use it to transfer files of various sizes between clients and servers at different distances away. (I.e. using different relays, or no relays at all.) Choose file sizes from 1 byte (overhead only) to files big enough to take approximately one minute to transfer. Report your findings in both a table and a plot. For this assignment your data analysis should expressly explore how the overall (effective, not counting duplicates) throughput varies in the face of increasing amounts of garbling.

Other Details:

- A makefile is required, that will allow the TA to easily build both your client and server. As always, you are free to develop wherever you wish, but the TA will be grading the programs based on their performance on the virtual computers provided by Amazon AWS for this class.
- MOSS will be used to check submissions not only against other submissions from this semester, but also against past submissions from previous semesters and any similar programs found on the web, so be sure to do your own work.

What to Hand In:

1. Your code, **including a makefile, and a readme file,** should be handed in electronically using Blackboard.
2. The intended audience for the readme file is a general end user, who might want to use this program to perform some work. They do not get to see the inner workings of the code, and have not read the

homework assignment. You can assume, however, that they are familiar with the problem domain (e.g. computer networking.)

3. A secondary purpose of the readme file is to make it as easy as possible for the grader to understand your program. If there is anything special the grader should know about your program, be sure to document it in the readme file. In particular, if you add any additional command-line parameters or special fields to the data packets or do any of the optional enhancements, then you need to document what they are and anything special the TA needs to do to run your program and understand the results.
4. If there are problems that you know your program cannot handle, it is best to document them in the readme file, rather than have the TA wonder what is wrong with your program.
5. Make sure that your name appears at the beginning of each of your files. Your program should also print this information when it runs.
6. The readme file must specifically provide direction on how to run the program, i.e. what command line arguments are required and whether or not input needs to be redirected. It must also specifically document any optional information or command line arguments your program takes, as well as any differences between the printed and electronic version of your program.
7. If there are problems that you know your program cannot handle, it is best to document them in the readme file, rather than have the TA wonder what is wrong with your program.
8. A printed copy of your program, along with any supporting documents you wish to provide, (such as hand-drawn sketches or diagrams) may be provided **to the TA** on the day the assignment is due or the following school day. The hard copies must match the electronic copy exactly.
9. Make sure that your **name and your ACCC account name** appear at the beginning of each of your files. Your program should also print this information when it runs.

Optional Enhancements:

It is course policy that students may go above and beyond what is called for in the base assignment if they wish. These optional enhancements will not raise any student's score above 100 for any given assignment, but they may make up for points lost due to other reasons. Note that all optional enhancements need to be clearly documented in your readme files. The following are some ideas, or you may come up with some of your own:

- Provide additional functionality, such as bi-directional file transfer, directory(tree) transfer, data compression, or encryption. See the manual pages for ftp( ) and other file-transfer programs for ideas.
- Anything else you can think of – Check with the TA for acceptability.