ASSESSED INDIVIDUAL WORK

G51CSF COURSEWORK 2
SOCKETS (ASSESSED)

Steven R. Bagley

Submission Deadline:
15/12/2017

Introduction

For this coursework, you are to implement a 'Quote of the Day' (QotD) server and client. This server returns a selected quote to any client that connects to it. There is a standard Internet protocol for this which is defined in RFC[1] 865 (a copy of which is attached to the end of this coursework). The RFC defines both a TCP and UDP version of the protocol but we are only implementing the TCP version. RFC 865 states that the server should listen on port 17 but since this is a privileged port **your client and server should use port 1717**.

The coursework is split into two parts. In the first part, you will create a server and a client that implement the standard QotD protocol defined in RFC 865 using C and the Berkeley sockets API. In the second part, you will implement a client and server for a modified version of the protocol which allows the user to fetch more than one quote during the session. More detailed descriptions of all four programs can be found below.

For both parts of the coursework, you can assume that only one client will be connected to the server at a time. An implementation of both servers is available that you can use to test your clients independently of your server, see the section on testing below for details…

Skeleton C files for all four programs are supplied via `git`, please ensure you use these for your implementation. It is expected that you are developing and testing this coursework on the school's Linux systems.

Quote of the Day Client

This program should connect to a QotD server on port 1717, read the quote sent by the server, and print it out to the standard output. The address of the server should be specified as the first parameter on the command line, e.g. like this:

```
$ ./qotdclient some.server.name
```

As specified in the RFC, you can assume that the server will not send a quote longer than 512 bytes. Once the quote has been read, you should print it out and close the connection to the server.

Hint: Remember that you might have to read from the network connection multiple times to obtain a complete quote.

Quote of the Day Server

This program should implement a QotD servers described in RFC 865.. It should listen on port 1717 for incoming connections from a client. When it accepts the connection it should then return

[1] Most Internet protocols are defined in documents called Requests for Comments (or RFC, for short)

a quote to the client. This quote should be presented on a single line, that is terminated with a carriage return (ASCII code 13) and line feed (ASCII code 10). You should ensure the entire quote (including `CRLF`) is no longer than 512 bytes (to comply with RFC 865). Once the quote has been sent, the server should close the connection and continue waiting for further connections.

A text file containing a series of quotes (one per line) is provided alongside the coursework in `git`, which your server can select a quote from. Your server should take the path to a quote file as a parameter on the command line.

Multi-quote of the Day Protocol

The second part of the coursework revolves around a modified version of the QotD protocol that allows the client to retrieve multiple quotes from the server. Instead of running on port 1717, this **multi-quote protocol runs on port 1818**, so you will need to adjust your client and server accordingly.

The server should respond with the first quote as usual. However, instead of closing the connection it should wait for a command from the client. All communication between the client and server is sent as complete lines that are terminated by a `CRLF`.

The command can either be a line containing the word '`CLOSE`' (in capitals), in which case the server should send the response '`BYE`' and then close the connection to the client. Alternatively, the client might respond with the command '`ANOTHER`' (in capitals) in which case the server should send another quote and await further commands from the client.

If the server receives any command from the client other than '`ANOTHER`' or '`CLOSE`', it should send the response '`ERROR`' and wait for a proper response from the client.

An example session between the server (**`S:`**) and client (**`C:`**) might look like this… (the **`S:`** and **`C:`** are for illustrative purposes only)

```
[C connects]
S: He's dead, Jim...
C: ANOTHER
S: One day I shall come back, yes, I shall come back
C: FRED
S: ERROR
C: CLOSE
S: BYE
[S closes connection]
[C closes connection]
```

Multi-quote server

The multi-quote server is very similar to the server you implemented before except that it should implement the revised protocol described above. The supplied quote file contains several quotes and it is perfectly acceptable to return them in the order that they appear in the file. As before, you should take the path to the quote file as a parameter on the command line.

Multi-quote client

The multi-quote client should take two parameters on the command line. The first parameter is the address of the server to connect to, as before. The second parameter is the number of quotes to fetch from the specified server, e.g. like this:

```
$ ./mq-client some.server.name 5
```

Your client should connect to the server and read the first quote from the server, before printing it to the standard output, as before. It should then send the necessary commands to the server, to cause the server to send the required number of quotes, printing each of them to the standard output, separated by a single blank line.

Hint: Again, remember that you might have to read from the network connection multiple times to obtain a complete quote.

Submission and Assessment

You should submit your four C programs via `git` in the usual fashion **by the 15th December**.

This coursework is worth 20% of the final mark, that 20% will be broken down as follows:

Quote of the Day Client	2.5%
Quote of the Day Server	2.5%
Multi-quote Client	5%
Multi-quote Server	10%

The marks will be awarded for each program based on whether it works, whether it implements the required features, whether it correctly implements the protocol as specified, and also the quality of that implementation.

Testing

To enable you to test your clients independently of your servers, there are test implementations of both the quote of the day server and the multi-quote server running on the machine `arrow.cs.nott.ac.uk` on ports 1717 and 1818 respectively. You can point your client at this machine to test your implementation before you have implemented your server.

Also, you might want to try testing the ‘interoperability’ of your client and server with your friends’ implementation, but remember your program should be your own work and so if you discover a bug it is up to you to work out what’s wrong with your program alone. You can find the address of your Linux session by using the `hostname` command and then connect to this from another Linux session.

Network Working Group
Request for Comments: 865

J. Postel
ISI

May 1983

Quote of the Day Protocol

This RFC specifies a standard for the ARPA Internet community. Hosts on the ARPA Internet that choose to implement a Quote of the Day Protocol are expected to adopt and implement this standard.

A useful debugging and measurement tool is a quote of the day service. A quote of the day service simply sends a short message without regard to the input.

TCP Based Character Generator Service

One quote of the day service is defined as a connection based application on TCP. A server listens for TCP connections on TCP port 17. Once a connection is established a short message is sent out the connection (and any data received is thrown away). The service closes the connection after sending the quote.

UDP Based Character Generator Service

Another quote of the day service is defined as a datagram based application on UDP. A server listens for UDP datagrams on UDP port 17. When a datagram is received, an answering datagram is sent containing a quote (the data in the received datagram is ignored).

Quote Syntax

There is no specific syntax for the quote. It is recommended that it be limited to the ASCII printing characters, space, carriage return, and line feed. The quote may be just one or up to several lines, but it should be less than 512 characters.