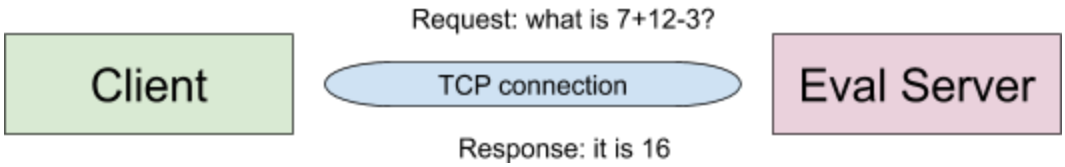
Programming Assignment 1

Goal

- Get familiar with writing simple networking code using socket API.
- Understand what is a protocol, and what is it for.
- Experience how to design application layer protocol.

Description

In this assignment, you will design and implement a service to evaluate arithmetic expressions, as well as a testing client. We use the simple client-server architecture, where a client sends server request with expressions to evaluate and a server response back with results. You need to implement using stream socket (based on TCP protocol).



Protocol Spec

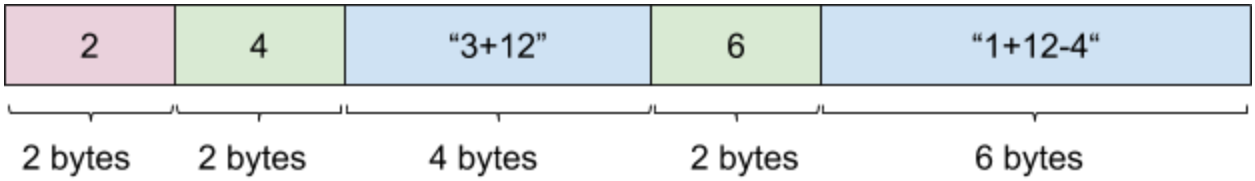
Because the service model provided by stream socket is a reliable stream of bytes (no meaning attached to the data, and no boundary on messages), we need to design an application layer protocol to define the syntax and semantics of the data we send between client and server.

Request format

Below is the specification for request message.

1. Number of expressions to evaluate. [2 bytes, encoded using network endianness]
2. Length of 1st expression in bytes. [2 bytes, encoded using network endianness]
3. String representation of 1st expression. [sequence of bytes]
4. Length of 2nd expression in bytes. [2 bytes, encoded using network endianness]
5. String representation of 2nd expression. [sequence of bytes]
6. …
7. Length of nth expression in bytes. [2 bytes, encoded using network endianness]
8. String representation of nth expression. [sequence of bytes]

Visualization of request format is shown in the picture below (note: quotation is only for clarity).

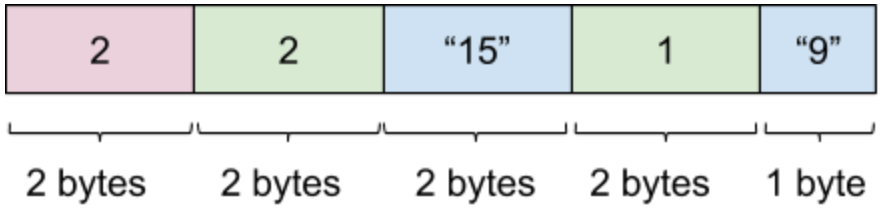


Response format

Below is the specification for response message.

1. Number of answers. [2 bytes, encoded using network endianness]
2. Length of 1st answer in bytes. [2 bytes, encoded using network endianness]
3. String representation of 1st answer. [sequence of bytes]
4. Length of 2nd answer in bytes. [2 bytes, encoded using network endianness]
5. String representation of 2nd answer. [sequence of bytes]
6. …
7. Length of nth answer in bytes. [2 bytes, encoded using network endianness]
8. String representation of nth answer. [sequence of bytes]

Visualization of response format is shown in the picture below (note: quotation is only for clarity)



Requirements

1. Stick with the protocol specification here, no modification in any way.
2. Use stream socket API (based on TCP protocol).
3. Eval server needs to be multithreaded, and can handle requests concurrently.

4. Expression to eval
    a. All numbers in expressions are positive integers.
    b. Eval server is only required to handle '+', '-' (don't need to worry about '*' and '/').
    c. No white space in the expressions.
    d. Assume all expressions in requests are valid.
    e. Don't use 'eval' in Python.
5. Implement a test client.
6. The `bufsize` used in recv API can not be larger than 16.

Common gotchas

There are two socket APIs where students often make the wrong assumption, which leads to incorrect implementation.

socket.send(bytes)

This function is used to send data to the socket. The `bytes` parameter holds the data you want to send. You cannot assume this function will send all the data at once. In fact, this function returns an integer, which is the number of bytes actually sent. You are responsible to check if all data is sent, and possibly call the function again to send remaining data.

Note: you are allowed to use `socket.sendall(bytes)` to make your life easier.

socket.recv(bufsize)

This function is used to receive data from socket. The `bufsize` parameter specifies the maximum amount of data to be received at once. You cannot assume `bufsize` number of bytes will be returned when you call this function, even if client/server has sent this much data (or more).

Grading policy

Grading based on the requirements above. 100 points in total.

1. Correctly use stream socket API, and create TCP connection (10 pts)
2. Limit `bufsize` to 16 bytes maximum and correct use of `recv` function (10 pts)
3. Format request and response correctly based on the protocol specification (30 pts)
4. Implement multithreaded server (10 pts)
5. Implement testing client (10 pts)
6. Pass 10 tests created by TAs (20 pts)
7. Coding style and necessary comments (10 pts)
8. Instruction on how to run your test client is recommended.

Submission instruction

Please your code and report to all TAs and cc z.sun@northeastern.edu by the deadline.