

Programming Assignment 1

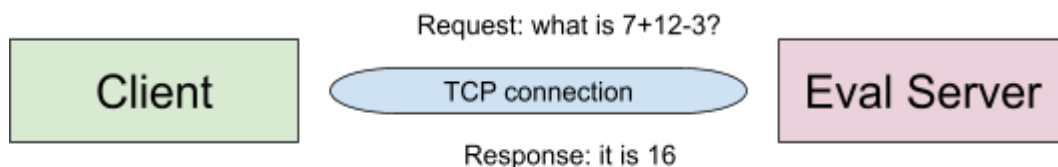
Goal	1
Description	1
Protocol Spec	1
Request format	1
Response format	2
Requirements	2
Grading policy	3
Submission instruction	3

Goal

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

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 a server request with expressions to evaluate and a server response back with results. You need to implement it using a stream socket (based on TCP protocol).



Protocol Spec

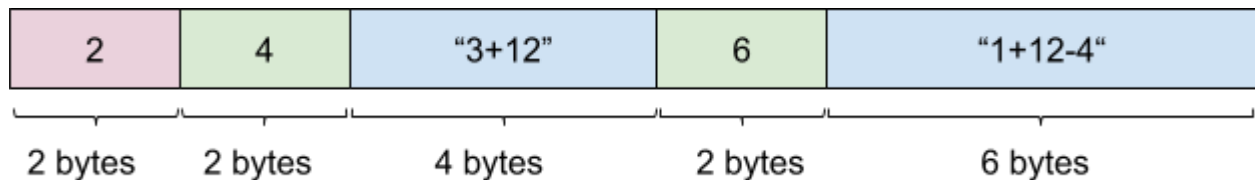
Because the service model provided by stream socket is a reliable stream of bytes (no meaning attached to the data, and no boundary on messages), we need to design an application layer protocol to define the syntax and semantics of the data we send between client and server.

Request format

Below is the specification for the 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).

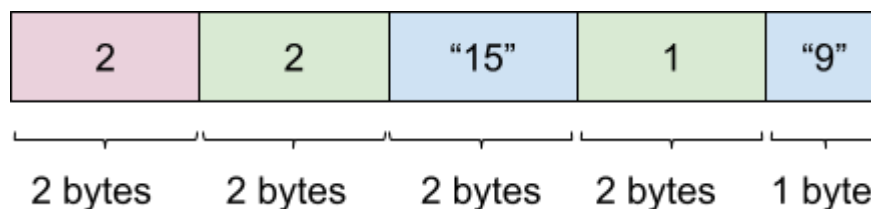


Response format

Below is the specification for the 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 '/', no '(' or ')').

- c. No white space in the expressions.
- d. **Assume all expressions in requests are valid.**
- 5. Implement a test client.
- 6. The integers in the protocol must be encoded with big endian
- 7. On Java implementation
 - a. Can only use java.io.**InputStream** to read from socket
 - b. Can only use java.io.**OutputStream** to write to socket
 - c. If max buffer size is 16 if using
 - i. `write(byte[] b, int off, int len)`
 - ii. `read(byte[] b, int off, int len)`

Grading policy

Assignment will be graded as per the following rubric,

Total points: 100

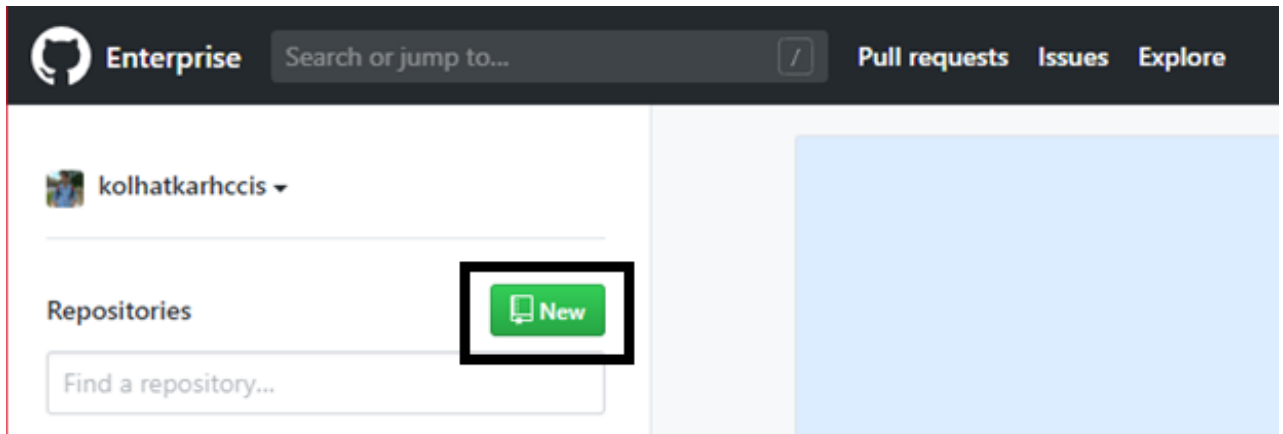
Points are distributed into 3 main categories and its further distribution is given below.

- Correctness of request/response message according to the protocol: 30 points.
 - Number of expressions field (with correct endianness): 10 points
 - Format of length of Expressions (with correct endianness) and the actual expressions: 20 points
- Implementation: 50 points
 - Handle concurrent requests using threads: 20 points
 - Use InputStream and OutputStream to handle read and write: 20 points
 - Max buffer size no more than 16: 10 points
- Expression evaluation: 20 points

Submission instruction

Submissions will be done by pushing the code to your own private repository on Khoury Github website. To create a private repository on the Khoury website, follow the instructions given below,

- Login to Khoury Github site using Khoury credentials. (<https://github.ccs.neu.edu/>)
- Once you login, you can create a private repo by clicking 'New' -> Give repository name



- Make sure that you select the repository as a Private repository.

A screenshot of the 'Create a new repository' form in GitHub. The form has a title 'Create a new repository' and a subtitle 'A repository contains all project files, including the revision history.' Below this, there are two input fields: 'Owner' (with a dropdown menu showing 'kolhatkarhccis') and 'Repository name' (with a text input containing 'TestRepo' and a green checkmark). Below these fields, there's a message 'Great repository names are short, simple, and unique. TestRepo is available. Need inspiration? How about expert-disco?'. There's a 'Description (optional)' text area. Below that, there are two radio buttons: 'Public' (unselected) and 'Private' (selected). Below the radio buttons, there's a section 'Skip this step if you're importing an existing repository.' with a checkbox 'Initialize this repository with a README' (unchecked) and a dropdown menu 'Add .gitignore: None'. At the bottom, there's a green 'Create repository' button.

- Once the repository is created, you need to add TAs as Collaborators to your private repo. You can do that by going to 'Settings' tab and selecting Collaborators on the left.
- In the search box you need to search for the following usernames and add them as collaborators. Usernames are 'adamseli' and 'kolhatkarhccis'

- Options
- Collaborators
- Hooks
- Notifications
- Integrations
- Deploy keys
- Custom tabs
- Autolink references


Collaborators Push access to the repository

This repository doesn't have any collaborators yet. Use the form below to add a collaborator.

Search by username, full name or email address

You'll only be able to find a GitHub user by their email address if they've chosen to list it publicly. Otherwise, use their username instead.

adamseli

 adamseli adamseli

Add collaborator