ENN523 Advanced Network Engineering

Group Assignment Part 1: Building a Client-Server System via Sockets

Version 1.0 on 21 March 2020

Distribution Date: Tue 22/03/22 (Week 3) **Due Date:** 11:59 Fri 29/04/22 (Week 7)

Assignment Submission

- Submit your assignment of a single compressed file via QUT's Backboard by the due date
- An extension of two days is granted automatically to every group for the submission

1. Introduction

This part of assignment on TCP/IP communications is a **group-of-two-or-three project** worth 15%. For special cases, individual attempt may be allowed subject to approval in advance from the unit coordinator.

Organisation of this document:
Sections 1 through 3 introduce some background relevant to this part of assignment. The assignment tasks are described in Section 4. Section 5 clarifies what you need to submit for your assignment. The marking scheme for the this part of assignment is enclosed in Section 6.

This part of assignment will require Socket Programming, which is the topic of the Lecture and Tutorial in Week 2. It relates to the following unit outcomes described in the unit outline:

2. Skills to undertake planning and design of computer networks to satisfy a set of requirements specifications with particular emphases on connectivity, scalability, reliability, security and QoS; and
4. Advanced collaborative and communication skills through a group project and formal technical report.

The criteria and performance standards used in this part of assessment are described in a table towards the end of this document (Section 6). Use the table as a marking guide.

You are asked to self-assess your assignment (in the enclosed marking guide table) and reflect on what you have achieved so far from study of this unit. This gives you the opportunity to reflect on what you have learned from this part of assignment and also what you need to further improve:

- For self-assessment, submit your self-assessed assessment sheet together with your assignment report and project source code. The self-assessment sheet is the marking guide table at the end of this document. It is provided to you in a separate and editable file.
- For reflections, write a separate section of Reflections in your assignment report. Each of your group members writes a separate paragraph of reflections specific to yourself.

Key technical aspects addressed in this assignment project include: *TCP/IP communications, socket programming, timing control, and round trip delay.* **No-technical aspects** of the assignment project include: *team work, report and communication, and reflections.*

2. Background: Dealing with Time

There are basically three types of methods to deal with time in C programming:

- Use some well-developed timing control APIs. For example, in Windows, a few functions are implemented in Windows library (header file: windows.h). More specifically, a function named QueeryPerformanceConter() is available for high-resolution timing control. In Linux (and MacOS), sys/time.h declares a few time functions for high-resolution timing control, e.g., gettimeofday().
- Use the standard time library from your operating system. The standard time library provides a number of functions for time operations, e.g., time(), localtime(), etc. Find a book or search the Internet to learn how to use the standard time library.
- Use hardware timer interrupt, which is the highest hardware interrupt. In this part of assignment, you are not required to use this method but you are free to use this method if you wish.

If anyone is interested in good timing control and clock synchronization over networks, our recent research papers on this topic will give you some ideas:

[1] Guosong Tian, Yu-Chu Tian, Colin Fidge, "Precise relative clock synchronization for distributed control using TSC registers", *Journal of Network and Computer Applications*, vol. 44, pp. 63-71, 2014. DOI: 10.1016/j.jnca.2014.04.013

[2] Khondokar Hasan, Charles Wang, Yanming Feng, Yu-Chu Tian, "Time synchronization in vehicular ad-hoc networks: a survey on theory and practice", *Vehicular Communications*, vol. 14, pp. 39-51, Oct 2018. DOI: 10.1016/j.vehcom.2018.09.001

[3] Fida Hassan, Yanming Feng, Yu-Chu Tian, "GNSS time synchronization in vehicular ad-hoc networks: benefits and feasibility", *IEEE Transactions on Intelligent Transport Systems*, vol. 19, no. 12, pp. 3915-3924, Dec 2018. DOI: 10.1109/TITS.2017.2789291

3. Background: Socket Programming

What is a "network socket"? A socket is a way for your application program to talk to other programs using the standard (Unix) file descriptors (Everything is treated as a file in Unix). A file descriptor is simply an integer associated with an open file; and the file can be a network connection!

Where to get this file descriptor for network communication? Call the socket() system routine. Then, send() and recv() and other socket calls can be used to talk to other computers. In this part of assignment, we deal with TCP socket only although other sockets are also available, e.g., UDP.

For a basic yet still comprehensive description on socket programming in C, refer to Beej's guide to Network Programming using Internet Socket (`http://beej.us/guide/bgnet/`). The guide is for unix/linux programming; however it also has a section for Windows programmers.

For more comprehensive descriptions of socket programing particularly in MacOS and Linux environments, refer to Glen's text "Building TCP/IP Socket" from Week 2 on the Blackboard. Glen's text also provides a section on windows socket programming.

4. Project Tasks

You are asked to develop **a Server** program running on one computer, and **a Client** program running on another computer. *When you test your server and client programs on one computer, you may use Loopback IP Address 127.0.0.1, with which any packets sent out from a computer will loop back immediately to itself.*

The architecture of the client-server system is shown in Figure 1. **The Server** will accept input from keyboard for system initialization, menu item selection, human command and instructions, etc. It will also display information on the monitor, periodically send commands to **the Client**, and receive feedback from **the Client**. After receiving a command from **the Server**, **the Client** will send feedback to **the Server**, and displays some information of your interest.

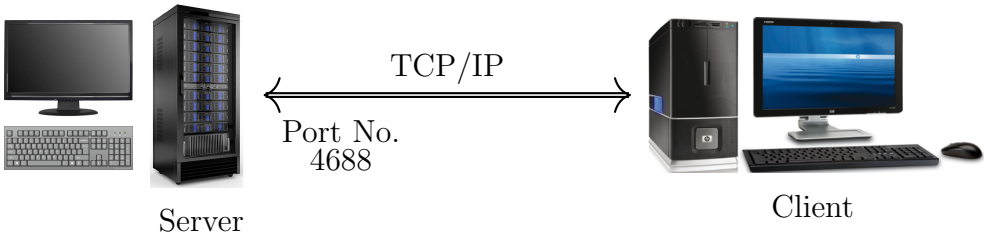

Figure 1: A client-server system including a server and a client.

Project tasks are described below:

1) When **the Server** starts to run, it initializes the settings of the client-server system, e.g., IP address and port number, through (i) command window arguments, e.g., header file, arguments to main(), (ii) keyboard input, or (iii) input from a configuration file, which is a pure text file.
2) Every three seconds, **the Server** sends **the Client** a command, e.g., through a single letter 'R' or 'r' (request) to request data from **the Client**.
3) After receiving the command from **the Server**, **the Client** responds to **the Server** with an ACK message consisting of:
    a) A string "ACK at" followed by a timestamp. The timestamp has the format hh:mm:ss:ddd, meaning two digits (hh) for hours, two digits (mm) for minutes, two digits (ss) for seconds, and three digits (ddd) for milliseconds (For example, 18:03:54:793, in which 793 means 793 ms); and
    b) A random integer in [0, 1000] with a uniform distribution. **The Client** may also display some useful information on its monitor. You may use a random generator function (available from C library) to generate such random numbers.
4) **The Server** gets the ACK message from **the Client**, and calculates the round trip time (RTT) from its communications with **the Client**, and displays the result on its monitor. *If you use the same computer to test the Server and Client programs, or if the Server and Client are very close, the RTT may be less than 1 ms. In this case, you will need a higher time resolution, e.g., in microseconds or even sub-microseconds.*
5) Without interruptions to the client-server communication, **the Server** reads keyboard input of various commands or instructions. An obvious command is to terminate **the Server** program, e.g., using a single letter 'E' or 'e' (exit) for termination. A three-way handshaking is required for terminating the client-server system:
    i) When **the Sever** is to be terminated, it should notify **the Client** of the Server's intention of termination so that **the Client** also terminates properly.

ii) When **the Client** receives the single letter 'E' or 'e' (exit) from **the Server**, it sends back to **the Server** a command using a single letter 'O' or 'o' (OK).
iii) Then, **the Server** informs **the Client** of the receipt of the 'O' or 'o' command.

After this three-way handshaking, both the **Server** and **Client** are safe to terminate.

How to test your programs:

(1) If you test your programs on a single computer, use the Loopback IP address 127.0.0,1. Execute the Server program in one command window, and execute the Client program in another command window.
(2) If you test your programs in computer labs, you may execute the Server and Client programs on two different computers, which have different IP addresses.

5. What to Submit

Submit your assignment via QUT's Blackboard. Your submission is **a SINGLE compressed file** of the following items:

1) A file of your assignment report on your design/solutions/discussions and other aspects that you feel relevant.
2) Your self-assessed assessment sheet in a separate file; and
3) Your c/c++ source files, header file/s, and data file if any. Do **NOT** submit any other project files that are specific to your IDE.

Some requirements for your reports are listed below:

1) The report starts with a cover page, followed by an Executive Summary within one page, and Table of Contents. Then, the report has the main body text, and references if any.
2) The body text could be organized with the following components: background of the project (e.g, main requirements and functions, etc), system design and logic flows at a high level (which is independent of any programming languages), implementation of the main components of the system (but not line-by-line code), test plan and testing results, reflections, conclusion, and references if any.
3) The report should be less than 20 A4 pages for the main body text. Additional materials that you feel important could be organized in an appendix. Note that do **NOT** include your code in the report.
4) Include a section of Reflections in the main body text to discuss what you have learned from this unit or this part of assignment, what you feel is beneficial to you, and what aspects you think you need to improve. Each of your group members writes a separate paragraph of refections; and an additional paragraph for the group is optional. Be specific about yourself and the unit/module/assessment, **NOT** be general about others.

6. Marking Guide

A marking guide is enclosed on the next page. It is also provided in a separate and editable spreadsheet, which will convert your raw marks (out of 100) for each assessment element into weighted marks (out of 15). Use this editable spreadsheet as your self-assessment sheet.

ENN523 Advanced Network Engineering: TCP/IP Communication Assignment, 2022 Semester 1. Marking Guide and Self-Assessment Sheet

Student1 Name: Student1 No.:

Student2 Name: Student2 No.:

Student3 Name: Student3 No.:

Element	Wt %	High Distinction 100-85	Distinction 84-75	Credit 74-65	Pass 64-50	To be improved 46-20	Your self-assessment		Marker's assessment	
							Raw marks (out of 100)	Weighted marks	Raw marks (out of 100)	Weighted marks
A. TCP/IP Server & Client	7	All services & commnucations functional with keyboard input to the Server. Use configuration file for initiatialization (IP addr., etc)	All reqervices & communications functional with keyboard input to the Server. Use keyboard input in initialization (IP addr., etc)	All services & communications functional with keyboard input to the Server. But all hard coded initialization (IP addr., etc)	Communications between Server and Client established with functional keyboard input to the Server. Other parts not all functional	Server or Client not functional		0 /8		0 /8
B. Report & assignment submission	5	All required (but no excessive) files provided in the specified format. Report professionally organised & presented	All required (but no excessive) files provided in the spoecified format. Report well organised & presented	All required (but no excessive) files provided in the specified format. Report well organised & readable	All required (and possibly exceesive) files provided; Readable report	Not all required files provided, or report organisation and presention need significant improvement		0 /5		0 /6
C. Group work	2	Group work, or individual with pre-approval	N/A	N/A	N/A	Individual without pre-approval		0 /1		0 /6
D. Self-assessment	1	Self-assessed, AND well written reflections	N/A	N/A	Self-assessed but assessment not reasonably done, OR simple reflections	No self-assessed, OR no reflections		0 /1		0 /1
Total	15	N/A	N/A	N/A	N/A	N/A	Overall	0 /15	Overall	0 /15
Overall Comments:										