ENN523 Advanced Network Engineering

Assignment Part 1: TCP/IP Communications via Socket Programming
Version 1.1 on 1 April 2021

Distribution Date: Thursday 8/03/21 (Week 2). **Due Date**: 11:59pm Friday 23/04/21 (Week 7)
Submit your assignment via QUT's Blackboard/Assessment/Assignment-Part1/
Academic dishonesty is not tolerable at QUT. Refer to MOPP C/5.3 on academic Integrity.

V1.1 Changes to v1.0 (All changes are highlighted in yellow in the text):

1) Page 3: to avoid confusion, in the format of communication messages, "seq#" is renamed to "seqno" to represent "sequence number" as in TCP protocol;
2) Page 3: in the format of communication messages, clarify that "ddd" means three digits to represent milli-second; and
3) Page 4: at the end of Section 6, indicate that "any programming language of your choice is fine".

1. Introduction

This assignment is designed as a **group-of-two-or-three** project worth 15%. Due to the COVID-19 pandemic, physical access to the university's computer labs is limited and a few students are studying off-shore. Therefore, **individual attempt** to this assignment is also allowed.

This assignment project on TCP/IP communications via socket requires Socket Programming, which is the topic of the Week02 Lecture and Tutorial.

Organisation of this document: Sections 1 through 3 introduce some background information relevant to this assignment. Assignment tasks are described in Section 4. Section 5 highlights what and where you need to submit. The marking scheme of the assignment is enclosed in a separate excel .xlsx file. It is a table of criteria and performance standards used in this assignment. Use it as a marking guideline.

This assignment 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.

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 assignment project and also what you need to improve.

- For self-assessment, submit a separate file of **your self-assessed assessment sheet**. Your assignment report should be in another file. Both files must be **in pdf format**. The self-assessment sheet is the table at the end of this document.
- For reflections, writes a separate section of Reflections in your assignment report. Each of your group members writes a separate paragraph of your **reflections specific to yourself**.

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

2. Background: Dealing with Time

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

(1) Use some well-developed timing control APIs. For example, in Windows, a few functions are implemented in windows library (header file: windows.h); and QueryPerformanceCounter() can be used for high-resolution timing control. **This is the method we have used in our examples. It is recommended to you for this assignment project** if you use windows.

(2) Using the standard time library. 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 this time library.

(3) Using hardware timer interrupt, which is the highest hardware interrupt. In this assignment, you are not required to use this method as it is more complicated.

For Linux users, sys/time.h declares a few time functions for high-resolution timing control, e.g., gettimeofday().

Time synchronization between two computers is not a requirement in this assignment project. 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) 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 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 assignment, we deal with UDP socket only although other sockets are also available, e.g., TCP.

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 brief section for Windows programmers.

4. Background: Application-layer ACK for UDP communications

In TCP/IP networking, TCP is connection-oriented and thus reliable through a built-in ACK mechanism. In comparison, UDP is connectionless and this unreliable. Therefore, TCP has a higher overhead and thus a larger latency than UDP. Because of this reason, UPD is preferable in many real-time and low-latency applications. But it is an issue how to use UDP for reliable communications. This assignment project shows a possible way of enhancing the reliability of UDP communications. We implement an ACK mechanism in Application Layer. This means that reliable communications can be established through an application-layer design when UDP is used.

The basic idea is as follows: For every message sent out from a sender to a receiver, once the receiver receives the message, it will send an ACK message to the sender. Then, there are more options to proceed. For example, after receiving the receiver's ACK message, the sender may further send an ACK message to the receiver to confirm the receipt of the receiver's ACK message.

If the sender does not receive an ACK message from the receiver after a desired period of time, it should re-send the message to the receiver. This is a re-transmission mechanism. However, to simplify the project, re-transmission is not a requirement in your assignment project.

If anyone is interested in this topic, the following references from our group will give you some ideas:

1) Xiaohan Shi, Guosong Tian, **Yu-Chu Tian**. A reliable transport protocol for real-time control over wireless networks. *IEEE Australasian Telecommunication Networks and Applications Conference* (*ATNAC'2012*), Brisbane, Australia, 7-9 Nov 2012. DOI: 10.1109/ATNAC.2012.6398082.
2) Li Gui, **Yu-Chu Tian**, Colin Fidge. A conditional retransmission enabled transport protocol for real-time networked control systems. *The 36th IEEE Conference on Local Computer Networks (LCN'2011)*, pp. 235-238, Bonn, Germany, 4-7 October 2011.

5. Assignment Tasks

You are asked to develop a **Server** program running on one computer, and a **Client** program running on another computer. The **Server** and **Client** communicate with each other via **UDP**. (*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 this machine will immediately loop back to itself.)

The Server will accept input from keyboard for system initialization, selection of menu items, 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. The structure of the system is shown in Figure 1.

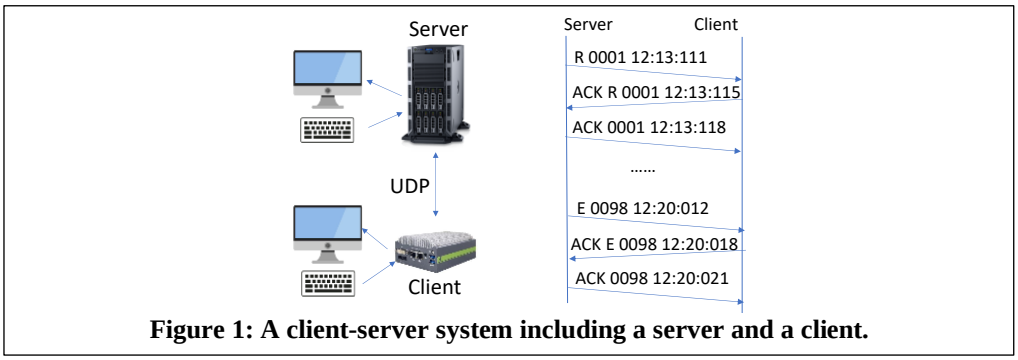

Figure 1: A client-server system including a server and a client.

Assignment tasks are described below (Refer to Figure 1):

1) When the **Server** is started, it initializes the settings of the server's IP address, the client's IP address, etc., through command window arguments, e.g., header file, arguments to main(), keyboard input, or input from a configuration file (which is a pure text file with initialisation settings).
2) Every 3 seconds, the **Server** sends the **Client** a command to ask for data, e.g., through a single letter "R" or "r" (request) followed by a space, a seqno, a space, and a timestamp in the form of hh.mm.ss.ddd, where "seqno" indicates a sequence number as in TCP protocol, "ddd" means three digits representing milli-second (e.g., 18:03:54:793). Therefore, the format of the request message looks like: "R 00001 12:13:58:386". You will need to control your sequence number of the message. (*Timing control is required here. Using our examples in the lecture materials if you like.*)
3) After receiving the request command from the **Server**, the **Client** sends back to the **Server** an ACK message starting with "ACK" followed by a space, letter R, a space, seqno, a space, and a timestamp in the form of hh.mm.ss.ddd. Therefore, the format of the ACK message looks like: "ACK R 00001 12:13:58:995". The **Client** may also display some useful information on its monitor.
4) The **Server** gets ACK message from the **Client,** records the timestamp at which the ACK message is received, and send a ACK message to the **Client** in the format of "ACK seqno timestamp hh:mm.ss.ddd" (e.g., "ACK 00001 12:13:00:011"). Meanwhile, the **Server** calculates the round-trip delay from its communications with the **Client**, stores the result, and displays the result on its monitor.
5) The **Server** reads keyboard input of various command and instructions. An obvious command is to terminate the **Server** program, e.g., using a single letter "E" or "e" (exit). When the **Sever** is to be terminated, the **Server** should also notify the **Client** of the Server's termination so that the **Client** also terminates properly by sending the **Client** an Exit message with a single letter "E" or "e" (exit) followed by a space, seqno, timestamp hh.mm.ss.ddd. So, the Exit message looks like: "E 0098 12:18:12:123". When the **Client** receives the Exit message from the **Server**, it sends back to the **Server** an ACK message in the format of "ACK E seqno timestamp hh.mm.ss.ddd (e.g., "ACK E 0098 12:18:12:256"). After receiving the ACK message from the **Client,** the **Server** sends an ACK message to the **Client** in the form of "ACK seqno timestamp" (e.g., "ACK 0098 12:18:12:302"). After that, both **Client** and **Server** will be terminated.

How to test your programs:

(1) *At early stages of your program development, you may test your programs on a single computer. Execute the **Server** program in one command window and execute the **Client** program in another command window. Both the Server and Client share the same IP address, e.g., the Loopback IP address 127.0.0.1.*

(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.*

6. Where and What to Submit

Submit your assignment via QUT's Blackboard/Assessment/Assignment-Part1. Your submission is a **SINGLE zipped file** of the following items: a pdf file of self-assessment sheet, a pdf file of report, and a folder of all source files:

1) Your **self-assessed assessment sheet** in an excel .**xlsx file**. It is the marking table provided in a separate excel .xlsx file for your easy marking and editing.

2) A **pdf file** of your assignment report on your design/solutions/discussions and other aspects that you feel relevant:

   a. The report should start with a cover page, followed by a declaration page, an Executive Summary of no more than 1 page, and Table of Contents. Then, the report has the main body text, and references if any.

      On the declaration page, declare the contributions from each of the team member if you work in a group for this assignment project. You are also required to declare that:

      - *I will uphold academic integrity as per MOPP C/5.3 and complete this assessment with academic honesty.*
      - *I declare that I have not been in contact with anyone who has divulged the contents of this assessment to me.*
      - *I declare that the answers I provide for the assessment are based on my own knowledge, and I have not been assisted in answering by any other student or person.*
      - *To ensure that assessment items are undertaken with the highest standards of academic honesty and to protect my integrity and the integrity of my QUT qualifications, I acknowledge that my responses to this assessment item may be verified by the Unit Coordinator or a delegate of the Unit coordinator.*

   b. The body text could be organized with the following components:

      i. background of the project (e.g, main requirements and functions, etc);
      ii. system design and logic flows at a high level (which is independent of any programming languages);
      iii. implementation of the main components of the system (but not line-by-line code);
      iv. test plan and testing results;
      v. analysis of round-trip delay performance;
      vi. reflections;
      vii. conclusion; and
      viii. references if any.

      The report should be less than 20 A4 pages for the body text. Additional materials that you feel important could be organized in an appendix. Do **NOT** include your code in the report.

   c. The report is expected to have **a section of Reflections** to discuss what you have learned from this unit or this assignment, what you feel is beneficial to you, and what aspects you think you need to improve. For reflections, each of your group members is expected to write a separate paragraph; and an additional paragraph for the group is *optional*. **Be specific about yourself and the unit/module/assessment, NOT be general about others**.

3) A **folder** of all your source files, header file/s, and data file if any. Source files ONLY, do **NOT** submit any project files.

   Note: Any programming language of your choice is fine for this assignment project though c/c++ is recommended as it is native to socket programming.

7. Marking Guide

Criteria Referenced Assessment (CRA) marking guide is enclosed in a separate excel .xlsx file, which is also used as a *self-assessment* sheet.