PROJECT SPECIFICATION Part 1
CS 4504 Distributed Computing

Due Date: See the Syllabus

Problem Statement

This is a group project. Each group will consist of three (3) students. Once your email addresses are determined, I will communicate with you the groups periodically. Alternatively, the Announcement and/or the Chat feature of D2L can be used to facilitate the communication within the group or use an online platform. Following is the specification of the problem.

Compile, install, and run the four (4) TCP program modules, which are in the Project Folder, to implement the following distributed system. In this system's configuration, there is a wired sub-network of regular PCs and a wireless sub-network of laptops with wireless network cards. The nodes in the two sub-networks communicate through a CS-Router (within the CS Department) and a Server/Router, S. The Server/Router is any ordinary PC in the CS-Lab or a laptop. The Server/Router S code also implements a routing table which is built into the program as a part of the simulation.

To illustrate the idea, consider the following scenario. Suppose that a node M is to send a request or message to a node N (either in the same sub-network or in the other sub-network) and node N processes the request and sends a response back to M:

1. First, node will M send a request to the CS-Router
2. The CS-Router directs the request to the Server/Router, S
3. Server/Router S's routing table is looked-up to find the destination address of the node N. (A routing table may contain the IP addresses, port #s and interface-link numbers, or some combination of these, plus any other pertinent information.) In this solution given to you, a minimum number of information is contained in the routing table.
4. Once N's address and port number are found in the table, the request or message is forwarded to it through the CS-Router. A reply from N follows the reverse path. (You need not worry about the CS-Router and, for wireless nodes, the KSU-UITS Department wireless router which will relay the message through the intermediate stages.)

In this distributed system each computer node is perceived as a client as well as a server. To further aid your understanding of the problem, in the following diagram, Figure 1, the nodes have *hypothetical* port numbers xxxx (e.g., 9999) and the interfaces of the nodes are numbered to facilitate the mapping. You need to determine the corresponding IP-address using finger, nslookup, or ipconfig, or similar protocol for the clusters of PCs you select in the CS-Lab in order to generate the needed sockets in your program modules. In the given code for Part 1, these numbers and/or the symbolic names were either fixed (as strings) or hard-coded as IP-numbers or port numbers. For example, the S-Router (in the SThread code) has a fixed port number 5555. The accompanying User-Guide describes how, in this case, the addressing information is entered and used in the code. You can change these if you want.

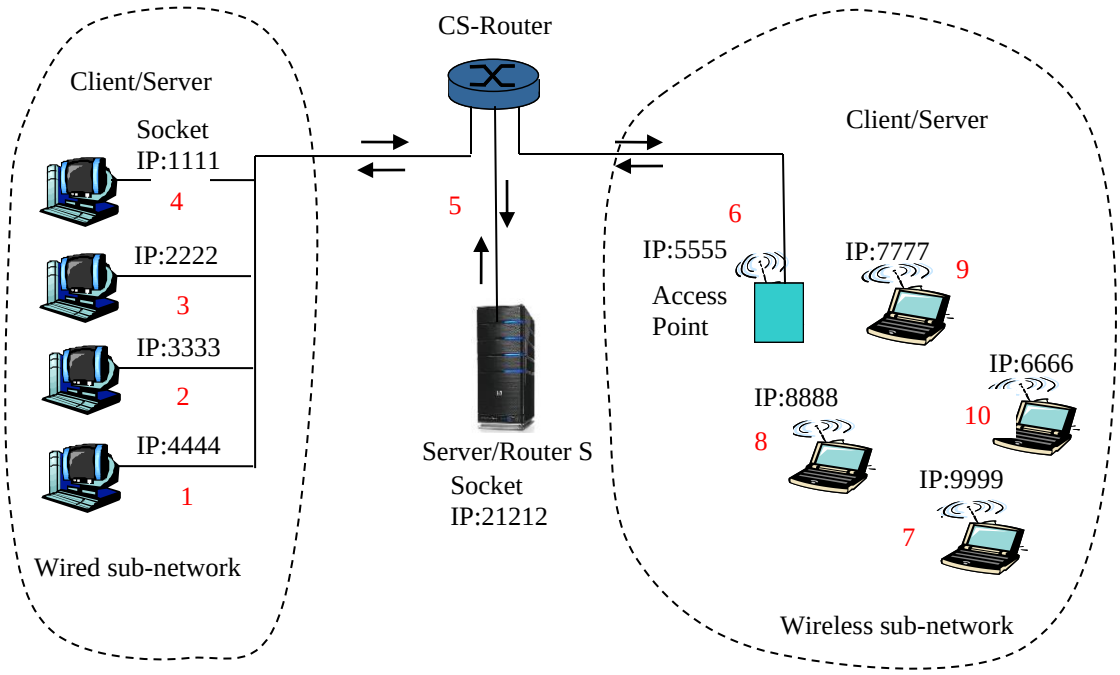

Figure 1: A conceptual network of two subnets (a wired and a wireless)

What to Do and Turn In

Complete source code for the client-side, the server-side, the server-router side, and threading code are given to your group. Study these programs as a group using the inline comments and code walkthrough techniques. Compile, install, and run the four (4) TCP program modules. Select as many nodes as possible on both the Client and Server sides of the network to simulate a fairly good size solution to the problem. (See Figure 1) Having a large size network also makes the simulation results statistically significant.

Test your system with several strings/messages, audio and video files of your choice. Each text file must exist on any node which is designated as a Client. Make slight changes to the code segments, wherever necessary, to collect and analyze data for such parameters as *average message sizes, average transmission time, average time for the routing-table lookup*, and any other of interest, if not already built into the given code. Write a 5-10-page report of your findings including 1) tables of collected data from the simulation and statistics and 2) analysis and conclusions from the data. The servers (in either side) are coded to convert each lower case to uppercase in the case of strings/messages. Use several text files of varying sizes and content for variability. (See the sample Text-file in Latin.)

After testing your code with text-file, modify the code to verify successful transmission of audio and video files, by playing back transmitted audio or video (using the appropriate AV application) on the receiver/server side. Again, use several AV files of varying sizes and content for variability. This Part 1 is intended to be completed in two-to-three (2-3) weeks, working in teams of three. (See the sample User-Guide.) **Follow the Rubric for submitting your final Report and a User-Guide guidelines for running your simulator**.

What Next?

Having completed and understood an implementation of the Client-Server paradigm using TCP, we have demonstrated that any two clients can communicate using a server-router as a bridge. The threading capability allows as many nodes as possible to be spawned for dynamic configurability and scalability of the network. In Part 2, which is based on the Peer-to-Peer (P2P) paradigm, will be implemented for the next five-to-six (5-6) weeks thereafter. (See the Part2 specs file.) The goal of this mini-project is help reinforce your understanding of these distributed computing (programming) paradigms and protocols.