# 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 five-to-seven (5-7) students. Once your D2L email addresses are determined, I will communicate with your groups periodically. Following is the specification of the *first part* of the semester course project.

Compile, install, and run the four (4) TCP program modules, which are in the PROJECT-FILES Module in D2L, to implement the following (simulated) distributed Client-Server (C-S) 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 of yours. 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) where node N process then processes the request and sends a response back to M:

1. First, node M will send a request to the CS-Router
2. The CS-Router directs the request to the Server/Router, S (the code on a PC or laptop)
3. Server/Router S's routing table is looked-up to find the destination address of the node N. (Generally, a routing table contains the IP addresses, port #s and interface-link numbers, or some combination of these, plus any other pertinent information, of all processes connected to a router.) In the initial Java code given to you, a minimum amount of information is contained in the routing table of the Server-Router S code. Add additional information as needed.
4. Once N's address and port number are found in the routing table, the request or message is forwarded to it (the destination) by the Server-Router S node. A reply from N follows the reverse path. (You need not worry about the CS-Router; it is automatically detected once a 'send() or receive()' API is executed in your code. Similarly, you need not worry about the wireless nodes since the KSU-UITS Department wireless router (or Access Point) will relay the message through the intermediate stages automatically.)

*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 and/or laptops you select in the CS-Lab 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 TCP-Server-Router code has a fixed port number 5555. Symbolic addresses need to be changed to the actual numeric IP addresses for the code to work.

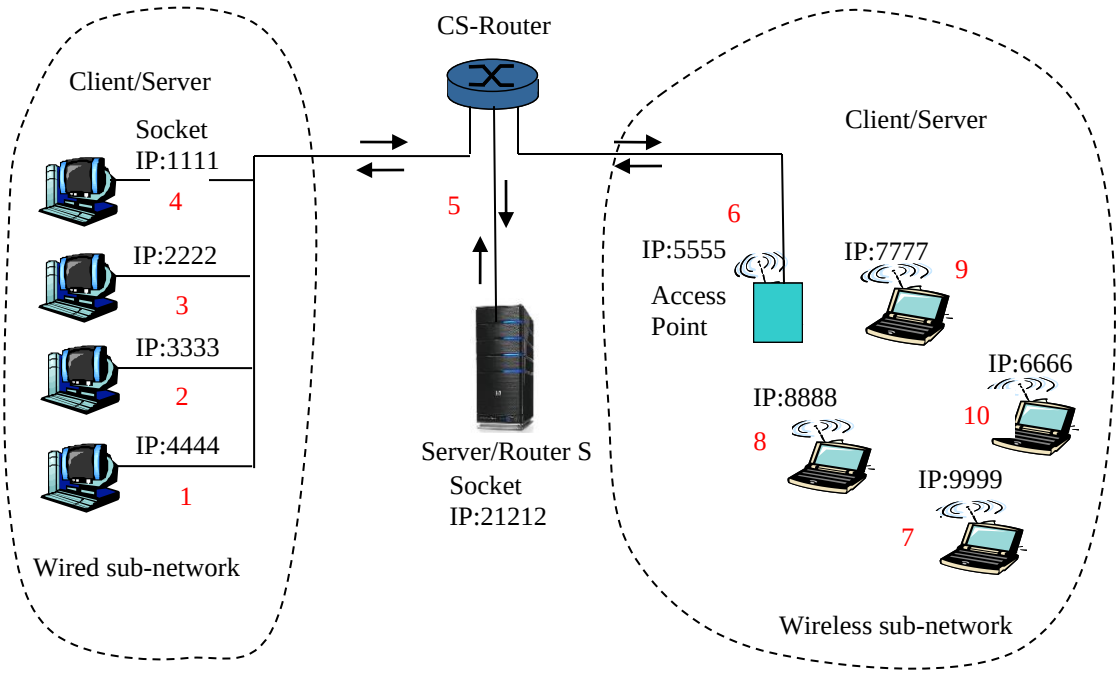


**Figure 1: A conceptual network of two subnets (a wired and a wireless)**

**What to Do**

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 inline comments and code walkthrough techniques. Compile, install, and run the four (4) TCP program modules following the data/control flow configuration depicted in Figure 1. 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. Having a large size network also makes the simulation realistic and the 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, 2) graphs corresponding to all the tables, and 3) analysis and conclusions from each set of data in the tables/graphs. The servers (on either side) are coded to convert each lower case to uppercase in the case of strings/messages. For audio and video files, the servers play back the content to verify correct transmission. (Note: The audio/video application or software must be installed on the server node for the playback to work.) Use several text files of varying sizes and content for variability; and varying sizes of the audio and video files.

Hint: Test your code with text-files first and collect all the simulation data/results. Then, modify the code to verify successful transmission of audio and video files, by playing back transmitted audio or video. Again, use several AV files of varying sizes and content for variability. Part 1 is intended to be completed in two-to-three (2-3) weeks, working in teams.

**What to Submit:**
Follow the Rubric in the PROJECT-FILES Module in D2L for writing, structuring, and submitting your final Report. In addition, *include as an Appendix to the Report*, a User-Guide – guidelines for setting up and running your simulator; and grading your project. (See the sample User-Guide in the PROJECT-FILES Module as a guide.)