Programming Assignment #1
— Constructing Peer-to-Peer Network —

Due on 11/08, 2011 (CSCI 4211), Max score: 100pt.

The objective of this project is to let you get hands-on experiences in computer network programming, in particular, socket programming, and in network application development.

In this programming assignment, you are asked to develop a simple peer-to-peer (p2p) network. You are asked to develop only one program, called 'peer'. Multiple hosts execute their own copy of this program in certain order (which will be elaborated later) to form a p2p network. We call such a host with a running 'peer' program as a 'peer host'. Every peer host designates a directory containing several files as its shared directory when it started. Our second project (p2p data sharing) will be built on top of this first project. Thus, it is important for you to implement this fundamental project well.

What you need to do?

You are provided with template source codes: peer.c, sockcomm.c and sockcomm.h. In peer.c and sockcomm.c, you can follow the instructions to fill missing parts. You are responsible for declaring additional variables and for adding some code blocks if necessary.

You are also provided with a compiled binary code 'peer'. It can be executed on Linux machines. You can follow the example in the later part of this description to see how the program is supposed to behave.

You are not restricted to work on the template provided. As long as your final program has the same function of 'peer' (including output messages), it will be acceptable. Any programming language is acceptable.

In order to compile your program, put files peer.c, sockcomm.c, sockcomm.h and Makefile under the same directory. Change your current working directory to that directory. Type command "make" and you will get binary code "peer". You can also use command line "gcc -lsocket -lnsl peer.c sockcomm.c -o peer" under Solaris, or use "gcc -lnsl peer.c sockcomm.c -o peer"under Linux.

If you don't have 'gcc' on your CS or itlab machine, that is, only because you haven't load the module yet. You can load the module using command 'module load soft/egcs/3.4'. In order to avoid doing this repeatedly, you can add this command line into your '.cshrc' file that is under your home directory.

Constructing a P2P Network:

The SYNOPSIS of the 'peer' program is as follows:

peer pathname [hostname]

- 'pathname' designates the shared directory of a peer host running this program.
- 'hostname' is optional. When it is not specified, the peer host has to be the first host in the p2p system. All others peer hosts have to specify 'hostname' in order to join the p2p system. Effectively, the p2p system grows like a tree as new peer hosts join the system, where the first peer host is root. There will be no loop within our peer-to-peer network.
- Example:
  Assuming I have compiled the source and generate the binary 'peer'. Before started, I copy 'peer' to my CS account home directory '/home/grad03/park'

  - Step 1: create directories on different hosts

    park@cello (~) % mkdir /home/grad03/park/cello
    park@saxophone (~) % mkdir /home/grad03/park/saxophone
    park@oboe (~) % mkdir /home/grad03/park/oboe
    park@trombone (~) % mkdir /home/grad03/park/trombone
    park@flute (~) % mkdir /home/grad03/park/flute

  - step 2: start peer host on 'cello'. Since it is the first peer in the system, that is root, no 'hostname' is specified.

    park@cello (~) % peer /home/grad03/park/cello

  - step 3: start peer host on 'saxophone', joining the system thru 'cello'

    park@saxophone (~) % peer /home/grad03/park/saxophone cello.cs.umn.edu

  - step 4: start peer host on 'oboe', joining the system thru 'cello'

    park@oboe (~) % peer /home/grad03/park/oboe cello.cs.umn.edu

  - step 5: start peer host on 'trombone', joining the system thru 'saxophone'

    park@trombone (~) % peer /home/grad03/park/trombone saxophone.cs.umn.edu

  - step 6: start peer host on 'flute', joining the system thru 'saxophone'

    park@flute (~) % peer /home/grad03/park/flute saxophone.cs.umn.edu

Joining the P2P Network:

In our system, a new peer host can join the existing p2p system from any present peer host in the system. Therefore, every peer host has to have a listen socket to accept connection request from new peers. In our program, every peer host has a listen socket on port '8000' for this purpose. A TCP connection is set up between the new peer host and its joining peer host. They are called a 'neighbor' to each other.

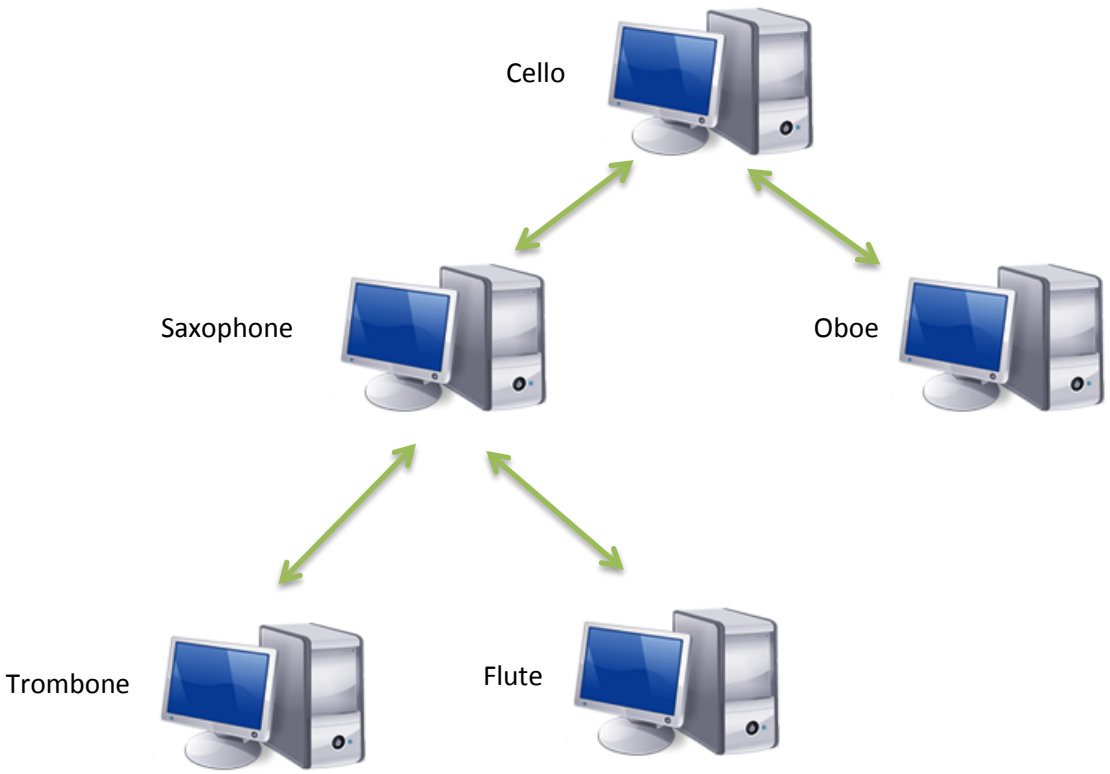

Figure 1: A Tree-based P2P Network

Terminate a Peer Host:

Use 'Ctrl-C' or ‘quit’ command to terminate a copy of 'peer'. It is required the termination to be handled gracefully, e.g., the connected neighbor(s) should display message like "admin: disconnected from 'saxophone.cs.umn.edu(35032)' ". Be assured that there is no crash on any neighboring peers.

Commands you must support in the peer program:

1. **quit**: to terminate a peer host.

Milestones:

● 10/13 (Thursday): Brief lecture on a socket programming and programming demo for your convenience.

● 11/08 (Tuesday): You must submit your **make file as well as all three files** (peer.c, sockcomm.c, and sockcomm.h).

► Please combine all those files into tarball (*.tar) or compress them for the convenience.

► File Naming convention: Last name + SID. tar (e.g. Park3341123.tar). Please follow this naming convention.

► Please use our on-line submission tool on the class website for your submission.