CSCI 6760 HWP 1: Locating peer nodes

Assigned: Tuesday, Jan 8
Due: Tuesday, Feb 5, 11:00AM
(LATE SUBMISSIONS WILL NOT BE ACCEPTED)

In this project, we will implement a scheme for locating and maintaining information about other peers that are currently online. There are two components to this project; first you will develop standalone peers that listen on a TCP port and maintain simple *key:value* tuples. Next, these peers will maintain information about other peers that are currently online and print the results on the terminal. For debugging purposes, the peers will continue to print location information such as peers entering and leaving the system.

1 Peers

The peers will maintain *key:value* tuples. The peers will listen in on a TCP port (of your choice). The peers will provide the following interface for services (note that the services are described in a 'C' like pseudo function call. You are free to implement it in a fashion that is convenient for you):

- **get(key)** This service will return the *value* associated with a given *key*. The key should be among the keys listed in the *list* service.
- **set(key, value)** This service will associate the *value* with the *key*. Existing values are overwritten with the new contents.
- **list()** This service will list all the keys that are available at the peer (set using earlier *set* operations).

1.1 Sample output:

We illustrate a sample interaction with a particular peer in greenhouse.cs.uga.edu:6003. Your interactions are illustrated in `typewriter font`.

```
$ telnet greenhouse.cs.uga.edu 6003
SET PinkFloydWall SongDataWillGoHere
OK SET
LIST
OK LIST
PinkFloydWall
GET PinkFloydWall
OK GET
SongDataWillGoHere
```

2 Location service

For this part, your goal is to identify other peers which are online. Peers identify each other by exchanging the *identification_t* structure. For this project, you will exchange the name, the Internet port number and Internet address where you can be reached in the *identification_t* structure. Peers will also maintain the round trip times to the different peers. You can refer to the COMPUTER NETWORKS book by *W. Richard Stevens* [2] for sample code on using network system calls. The sample code from this book is available online at http://www.kohala.com/start/unpv12e.html.

```
typedef struct identification {
  char name[32];                /* Name of the current client */

  // Specify how we can be contacted.
  in_addr_t  location;       /* IP address of the client */
  in_port_t  port;           /* port where the client is listening */
} identification_t;
```

There are a number of different ways to locate other peers. For this project, you will use peer-to-peer techniques to locate other peers (and not centralized approach). The peers will directly locate other peers without any centralized data structures. Peers can utilize multicasting (all peers listen on different multicast channels) or broadcasting to identify other peers. Peers can broadcast a query asking other peers to identify themselves or new peers can initially broadcast their identity in order to join the community. You should be able to locate instances of your own peer running on different hosts (you would have to explicitly start a number of peers). You may also be able to locate peers developed by your classmates.

2.1 Sample output:

This is a sample run for how you might print other peers entering and leaving the system.

1/9/2002 10:30 ‘John Doe’ ENTER gemini.cs.uga.edu:6780 20 msec

1/9/2002 10:35 ‘John Doe’ MAINTAIN gemini.cs.uga.edu:6780 30 msec

1/9/2002 10:36 ‘John Doe’ LEAVE gemini.cs.uga.edu:6780

1/9/2002 10:40 ‘Jane Doe’ ENTER greenhouse.cs.uga.edu:6003 100 msec

3 Implementation details

For this project, you will have access to 6 FreeBSD based machines (located in Boyd Rm 539). These machines are named after the 7 dwarfs (dopey, sleepy, sneezy, grumpy, happy and bashful). You can login to these machines by first ssh’ing to *tornado.cs.uga.edu*. You should test your implementaions on the different machines. Each machine is configured to simulate different network conditions using *dummynet* [1].

4 Submission

Submit your project, along with a succinct report called REPORT.txt (plain text is fine) describing your approach, the merits of your approach and compilation instructions. You will turn in your complete project as a single tar file. On gemini, please use `/home/profs/surendar/bin/turnin NETWORKS HWP1 <your tar file>` to submit your assignment. You can submit your assignment multiple times. I will only use the latest submission. To see the files that you had submitted, try `turnin NETWORKS HWP1`. **Remember, I will randomly choose students who will be asked to explain their approach in person**. Evaluate your implementation on the following issues in the REPORT.txt:

1. **interoperability:** How does your peer recognize your friend in a different host/operating system. For example, if you are working in gemini [Sun Sparc machine running Solaris], can you identify your friend in the dorm using a Pentium III running Linux?

2. **scalability:** If your peer system becomes wildly popular (ala napster), can your system handle tens of millions of peers?

3. **consistency:** How quickly do peers realize when a peer crashes so that the display that you get accurately reflects the peers that are currently online?

References

[1] Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, 27(1):31–41, January 1997.

[2] W. Richard Stevens. *UNIX Network Programming: Networking APIs: Sockets and XTI*, volume 1 of *ISBN 0-13-490012-X*. Prentice Hall, 2 edition, 1998. Sample code from this book is available at http://www.kohala.com/start/unpv12e.html.