CSCI 4900/6900 HW 1: Locating Beacons

Assigned: Tuesday, Jan 9
Due: Tuesday, Jan 23, 11:00AM (LATE SUBMISSIONS NOT ACCEPTED)

1 Motivation

Locating computing resources is an important first step in accessing ubiquitous resources. For example, when you walk into this class, you want to know who/what else is accessible. You can use this information to communicate with your "friends", find the closest printer etc. You can also use this information to allow your "friends" to communicate with you. For ubiquitous computing, location management is part of a larger problem of authentication/authorization/protocol negotiation etc. In this project, we will implement a simple scheme for locating other beacons.

2 Description

The goal of this project is locate beacons that are currently online. Beacons are defined as software representations of physical entities. Beacons can represent printers, scanners, LCD projectors, people, microwave ovens, etc. In this project, the beacons will represent you. Your goal is to identify others who are online at the same time. Beacons identify each other by exchanging the *identification_t* structure. For this project, you will exchange your name, the Internet port number and Internet address where you can be reached in the *identification_t* structure. You can refer to the Computer Network books by W. Richard Stevens [1] 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.

```
// Structure that is exchanged between clients to help identify the
// client location, how to contact the client and what protocol to speak

typedef struct identification {
  // Identify who we are
  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 */


  // Specify my credentials
  char key[32];                /* My authentication key. Unused for
                                  this project  */

  // Specify how to talk to us. Unused for this project
  unsigned int  type;          /* whether we talk XML, HTTP, Corba protocols */
  unsigned int length;         /* Length of protocol specific data
                                  that follows in buf */
  void *buf;                   /* Protocol specific buffer */
} identification_t;
```

In general, there are a number of different ways for beacons to identify other beacons. Some popular techniques are:

1. **Central Server:** This approach uses a centralized server. Every beacon registers itself with this server. One would query this central server to search for other beacons. You have to first know where this central server is located before you can ask it for the beacon locations (boot-strap problem). Some popular examples utilizing this approach include napster (www.napster.com), AOL Instant messenger etc.
2. **Peer-to-peer:** Here the beacons directly locate other beacons without any centralized data structures. Beacons can utilize multicasting (all beacons listen on different multicast channels) or broadcasting to identify other beacons. Beacons can broadcast a query asking other beacons to identify themselves or new beacons can initially broadcast their identity in order to join the community. Gnutella (www.gnutella.org) utilizes such peer-to-peer techniques.

For this project, you are free to choose any of the techniques described above or a technique of your own. Depending on the technique that you use, you should be able to locate instances of your own beacon running on different hosts (you would have to explicitly start a number of beacons). You may also be able to locate beacons developed by your classmates.

3 Submission

Please submit your project, along with a succint 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 hw1 <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 hw1`. **Remember, I will randomly choose students who will be asked to explain their approach in person**. Issues that you might consider while developing and evaluating your system are:

1. **interoperability:** How does your beacon 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 beacon system becomes wildly popular (ala napster), can your system handle tens of millions of beacons?
3. **consistency:** How quickly do beacons realize when a beacon crashes so that the display that you get accurately reflects the beacons that are currently online?

4 Sample output:

```
% beacon
1/9/2001 10:30 'John Doe' gemini.cs.uga.edu:6780
1/9/2001 10:35 'John Doe' LEFT
1/9/2001 10:40 'Jane Doe' greenhouse.cs.uga.edu:6003
```

References

[1] 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.