CSCI 4900/6900 HW 2: Distributed Services

Assigned: Tuesday, Jan 23
Due: Tuesday, Feb 6, 11:00AM (LATE SUBMISSIONS NOT ACCEPTED)

1 Motivation

In the last home work project, we located other beacons that are currently online and accessible. We know where the other beacons are and how to talk to them (the location and port fields of the *identification_it* structure). The next step is for the beacons to provide simple services. Depending on the object that a beacon is attached to, beacons can provide any number of different services. For example, a beacon attached to a printer can advertise the printers capabilities (postscript/PCL, color/greyscale, 1200dpi, inkjet etc), accept jobs for printing and provide status information (printer jam, out of paper, out of ink etc.). A more complex beacon can make requests to other beacons to accomplish its tasks. For example, the printer beacon can in turn contact another beacons to convert a powerpoint document to postscript format so that it can be printed. The printer beacon can also contact a clearinghouse to charge the user for the printing costs.

2 Description

The goal of this project is for the beacons to provide a simple service. For this project, the beacons will provide a service that will list and serve files. The beacon should be waiting for requests at the address and port specified by location and port values of the *identification_t* structure. The beacons 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):

- **open(identification_t:key)** All the requests to a beacon should be preceded by the open function. Beacons need a way to authenticate the user who is making a particular request. For our project, we assume that the key is always valid. The beacon will return a token that identifies a particular session. Further requests to this beacon should be accompanied with this token for service. A request without a valid token should be denied.
- **list(token)** This service will list all the files that are available at the beacon. You are free to choose the files that will be listed by this service. Your beacon should at least list 2 file names.
- **get(token, file)** This service will send the contents of the file requested. The file should be among the files listed in the list service. Requests for a file that is not available should be denied.
- **put(token, file)** This service will upload the contents of the file. Existing files are overwritten with the new contents.
- **close(token)** This service will end the session with this particular beacon.

Our beacon, in fact, provides a rudimentary service for file exchange that is similar to the services provided by systems such as gnutella, napster etc. As usual, you are free to choose the exact technique to provide the services described above.

3 Sample output:

The purpose of this sample output is to give a sense for the expected interaction with the beacon service. You are free to choose the scheme by which you will interact with your beacon. For this sample output, a $>>$ in the beginning of the line signifies that the beacon was waiting for user input.

```
% beacon
1/23/2001 10:30 ‘John Doe’ gemini.cs.uga.edu:6780
1/24/2001 10:35 ‘John Doe’ LEFT
1/25/2001 10:40 ‘Jane Doe’ greenhouse.cs.uga.edu:6003
>>open "Jane Doe"
DEBUG: Opened Jane Doe (greenhouse.cs.uga.edu:6003) -> key = 04596F
>>list 04596F
Metallica.mp3
DrDre.mp3
hw2.pdf
>>get 04596F hw2.pdf
DEBUG: Saving to file hw2.pdf
>>close 04596F
DEBUG: Session close completed
```

4 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 hw2 <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 hw2`. **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. **robustness:** How reliable is your system against failures? If a beacon crashes and came back online immediately, can your system continue to provide service to existing clients?

2. **scalability:** If your beacon suddenly becomes popular because of the files that your provide, how much load can you tolerate before your system crashes? Does your service degrade gracefully? Are you immune to denial-of-service attacks? (wherein, beacons repeatedly open connections to you to prevent you from servicing other, legitimate users)

3. **security:** How secure is your system regarding key management? If I open a session, received a token and gave it to a friend, would they be able to utilize the key to access files? Can I receive files that were not listed by the **list()** service?