CSCI 6760 HWP 2: Application level query routing among the peers

Assigned: Tuesday, Feb 5
Due: Tuesday, Mar 5, 2:00 PM
(LATE SUBMISSIONS WILL NOT BE ACCEPTED)

In the last home work project, we used peer-to-peer technologies to locate other peers that are currently online and accessible. We also implemented a simple service that manages *key:value* tuple pair. The next step is to be able to access keys that are available in peers that are not directly known to the current peers.

For this project, the peers from HWP1 will be restricted to manage location information about **two** other peers. When contacting remote peers, you are only allowed to directly contact a peer that you already know about. All other peers should be queried through peers that you know about. For example, suppose we know about peer FOO on host1:port1 and we get a request for BAR on host1:port1, we have to send the request to FOO which might forward the request to BAR. The route management policies should use the RTT times (measured in HWP1) to choose the 'best' peer. We extend the services provided by Home Work 1 as follows:

- **search(searchKey, hopCount)** If the requested key *searchKey* is available in the peer, the identity of the peer (in a printable *hostIP:port* format) is sent back. If the key *searchKey* was not available, a recursive **searchget** is invoked by this peer (on behalf of the requestor) on all the peers that it knows of (restricted to two for this project). Every such forwarding decrements the hopCount. Once the hopCount reaches 0 without successfully finding the file, the system returns an error message.

  Note that your implementation might return multiple values for the same key. It might also return an KeyNotFound error, even though there is a path available from the source to the destination.

- **rget(peerHost, peerPort, key)** This service will return the *value* associated with a given *key* in the peer at peerHost:peerPort. You are only allowed to directly contact the peer at peerHost:peerPort if your own internal routing tables know about this peer. If you do not have direct knowledge of the peer, you should forward it to a peer that you know.

As usual, you are free to choose the exact technique to provide the service described above.

Submission

Please 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 HWP2 <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 HWP2`. **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? Does your peer recognize forwarding loops? (wherein the requests are forwarded around in a loop without making progress towards the destination)

2. **scalability:** If your peer 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, peers repeatedly open connections to you to prevent you from servicing other, legitimate users)