Peer-to-peer networks – (due till April 22, 2009)

Exercise 6.1:

In Chord, a node forwards a request to a neighbor which was addressed by what we called the i-th finger. Implement the function (in java, C or pseudocode) with the signature given below. It should return the best fitting neighbor (don't overshoot the mark, we can not go back easily) from the list `successor[]` who is best suited for the key `hash_value`. You can assume that our hashes fit into a long integer. Our own ID is: node.ID

```
long returnFinger(long successor[], int no_entries, long hash_value);
```

Solution 1:

```
long returnFinger(long successor[], int no_entries, long hash_value)
{
     long best_i = -1;

     long relative_hash = (hash_value – node.ID);
     if(relative_hash < 0) relative_hash = (1 << no_entries) + relative_hash;

     for(int i = 0; i < no_entries; i++) {
        if(relative_hash >= (1 << i))
           best_i = i;

     } // for

     return successor[best_i];
} // returnFinger
```

Peer-to-peer networks

Exercise 6.1:

Solution 2:

```
long returnFinger(long successor[], int no_entries, long hash_value)
{
     long relative_hash = (hash_value – node.ID);
     if(relative_hash < 0) relative_hash = (1 << no_entries) + relative_hash;

   long best_i = (int)(log(relative_hash)/log(2));

   return successor[best_i];
} // returnFinger
```

Peer-to-peer networks – (due till April 23, 2008)

Exercise 6.2: Arrival of a new node in Chord

a) In the lecture, we learned how the Chord protocol manages the key space and how routing is done. We also saw a brief outline of the insertion of a new node. The text

chord.pdf

on our homepage gives further details on how the insertion is actually implemented. Describe the soft-state approach adopted by the designers of Chord.

Solution:

A new node N queries the network for its own ID and gets its successor S as a result. It then chooses S for its 2^0 finger and queries the nodes for all other finders. In addition, N asks S to register him as his predecessor. So far, no one else except for S knows about N. As a consequence, all nodes in the network will route to the well-known S and not to N. However, S can route the query back to N via its predecessor pointer.
Once in a while, all nodes query the network for their finger entries and sooner or later learn about the existence of N.

Peer-to-peer networks

Exercise 6.2: Arrival of a new node in Chord

b) Can you think of advantages and disadvantages of the soft-state solution?

Solution: Since every node has to update every finger periodically, the effort for the network is log(N x log(N)) every once in a while. The effort gets worse with a decreasing time between the updates. A longer time means outdated finger entries and possibly more effort for the forwarding along the predecessor-pointers. The frequency of updates depends heavily on the rate of joining and leaving nodes which are not easy to anticipate and which will not be constant over time.

c) Find a more deterministic solution which does not build on top of a random choice.

Solution: A node t could store a backward-pointer to every node that ever included t into its finger table. When getting a predecessor, t could inform these nodes about the change. They can then decide to keep their reference to t or rather to point a the predecessor if appropriate. Still unsolved problem: If t drops out, other nodes keep pointing at t. The soft-state approach solved this problem with the continuous updates.