CS347 Homework #5 (Peer to Peer Systems)
Due: Wednesday May 26, 2010 (5pm)

Problem 1

Consider a Chord system with $m = 7$ (address space goes from 0 to 127). Nodes whose ids hash to the following values have joined: 15, 35, 55, 75, 95, 115. Show the finger table for node 35. Use a table analogous to the one in Slide 22 (in P2P Notes).

Problem 2

Consider a Chord system with nodes with ids that hash to 1 and 10. At the same instant of time, three nodes join: one has id that hashes to 2, another one hashes to 4 and the third hashes to 7.

(a) Assume that after the three nodes join, they execute their periodic stabilization (slide 34, P2P Notes) in the order: 7, 4, 2, 1. Assume that after each stabilize call, the corresponding notify call is made and completed before the next stabilized call is made. Show the resulting pred and succ links for each node (after the 4 stabilize calls complete, and before any other calls) in a diagram like the one in slide 38 (P2P Notes). Use solid arrows for the succ links, dashed arrows for the pred links.

(b) Same question as part (a) except that the stabilize calls are in the order: 2, 4, 7, 1. Again, show the resulting links in a diagram.

Problem 3

(a) Slides 72–78 (P2P Notes) show the seven steps executed when a node joins a replicated hash table. For each of these steps (slides), state what happens to lookups for keys with hash 0 executed at N0, when the state is as shown. (In particular, is the lookup executed locally? Is it forwarded to some other node?) Then state what happens to lookups (again for keys with hash 0) executed at N1. Finally, explain what happens to N2 lookups.

(b) Same question as part (b) except that you should now state what happens to inserts.

Problem 4

Briefly explain how the replicated hash table described in class (slides 66–82, P2P Notes) can be extended to use an extensible hash table. As with extensible hash tables (covered in CS245), we use the "i" high order bits of a hash value to look up a value in a hash table. However, now each hash table can use a different "i" value since we do not have a central controller. Thus, we need to be careful when a node received a message from a node that has a different "i" value than its own.