Chapter 1

Evaluation

1.1 Experimental design

For the purpose of this section, *host* refers to a physical machine, while a *node* is an instance of a distributed hash table. Each host will run one or more nodes each.

The aim of this evaluation, and of this whole dissertation, is to determine how well the distributed hash tables are suited as data stores for my friend search engine. This influences the experimental design, which will focus on testing the practicality of using the distributed hash tables as back end stores and mainly focus on different aspects of resource consumption.

1.1.1 What will be tested

Three separate experiments will be performed:

1. The time it takes to perform a lookup
2. The number of nodes involved in a lookup
3. The amount of bandwidth required to do a lookup
4. The amount of bandwidth required to maintain routing state

Each experiment will be performed for both my implementation of chord and pastry and for different number of nodes.

The experiments will follow a factorial design, where factors are:

- Number of nodes in the network

- Rate of requests

In each of the experiments, given N hosts, the number of nodes will be 2^kN, where k is the increment level ranging from 0 to 3 in each experimental run. Since the number of nodes involved in any lookup is thought to be proportional to the logarithm of the number of nodes in a given network, this seems like it should give interesting data to analyse.

The time it takes to perform a lookup

This is a very interesting experiment in terms of end user experience. Obviously we would want the response time to be as low as possible. It will be interesting to see how the response time is affected by both the size of the network and the rate of queries. To produce a balanced load, each host will issue requests to the network.

The factors are network size and the rate of requests. For each level of network size, each host will issue requests serially at a certain level of parallelism. By level of parallelism I mean that at level 1, a host issues requests serially, at level 2 it will have two requests in flight in at any one time, etc. The level of parallelism increases by 1 every minute as long as no host sees a failure rate of more than 25

The amount of runs required for this experiment to be complete, depends on how well the network copes under load.

Number of nodes involved in a lookup

This experiment focuses on how large a fraction of the network has to be involved in a request and how this scales with the size of the network. This is relevant as the number of nodes involved tell us something about how well the network will manage under heavier load. For this experiment the rate of requests is irrelevant, and likewise if the requested value is present on the other host or not. For this experiment the only factor is network size, and the response variable the number of nodes involved in a lookup. For one run of this experiment to be completed, we need data for the number of lookups for each of the 4 possible levels of the number of nodes factor for each of the distributed hash tables.

The amount of bandwidth required to do a lookup

This is an experiment that is interesting in terms of real world applicability where the nodes would be hosted by end users paying for the bandwidth consumed by their servers. The actual bandwidth will naturally be implementation specific, but will still give us a picture of how chord and pastry compare. This experiment will be performed as part of the experiment counting the number of nodes involved in a lookup.

The amount of bandwidth required to maintain routing state

This is also an experiment that very much helps determine if the implementations are usable in a real world scenario. The bandwidth consumption will depend on the configurable options of chord and pastry. It therefore becomes hard to do a fair comparison based on the difference of these values. Values will therefore be chosen that generally prove to give good performance.

1.1.2 What will not be tested

The behaviour of the networks depends on their different configurable options. In chord you can adjust the number of successors it keeps in touch with, and the frequency at which these relations and the routing table is kept up to data. In pastry in addition to choosing the rate at which the routing tables are maintained, there is a *b-parameter* influencing how many routing steps will be required but also the size of the routing tables.

Since the configurable parameters do not directly correspond between the two implementations, their levels will be fixed at values that generally give good and consistent performance.

While both my implementation of chord and pastry replicate their data for fault tolerance, this behaviour will not be tested. It is expected that the network size will not fluctuate wildly over time in a real world deployment, as the nodes likely will be hosted on servers expected to be in operation over long periods of time. This is very much unlike other distributed hash table usage scenarios like file sharing where the amount of nodes joining and leaving the network at any given time can be significant (I don't have any data to back this claim with, but it sounds reasonable!?).

1.2 Experimental results