Chapter 1

Evaluation

1.1 Experimental design

In this section, *host* refers to a physical machine, while a *node* is an instance of a distributed hash table running on a host. Each host will run one or more nodes.

The focus will be on testing the practicality of using the distributed hash tables as back end stores for my friend search engine, and mainly focus on different aspects of resource consumption.

1.1.1 What will be tested

Four 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 separatly for chord and pastry, in each case for networks of different sizes.

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

- The number of nodes in the network
- The rate of requests

In each experiment, given N hosts, the number of nodes will be 2^kN — k taking levels 0 through 3 per experimental run. As the number of nodes involved in any lookup is thought to be proportional to the logarithm of the number of nodes in a network, this seems likely to give interesting data to analyse.

None of these experiments rely on querying for real data. Keys to lookup will therefore be generated by hashing a combination of a counter value and the identity of the host performing the request. This is likely to give an even spread of keys.

The time it takes to perform a lookup

This is an interesting experiment in terms of the expected end-user experience. We want to observe how the response time is affected by network size and request rate. The outcome of this experiment dictates how complicated queries the network can sustain, and more concretely if predictive searches in its current form can be sustained on a larger scale. Each host will individually issue request to produce a balanced load.

The factors are network size and the rate of requests. For each level of network size, each host will issue requests serially at increasing levels of parallelism until any one host sees a failure rate of more than 25%. Each level of parallelism will be kept for a minute.

The number of runs required for this experiment depends on how well the networks cope under load.

Number of nodes involved in a lookup

This experiment focuses on what 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 can indicate how well the network will manage under heavier load. The request rate is irrelevant 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, we need data for each of the 4 possible levels of network size.

The amount of bandwidth required to do a lookup

This experiment is interesting in terms of real world applicability. The nodes are likely to be hosted by end users paying for the bandwidth consumed by their servers. The actual bandwidth consumption will to a certain extent be implementation specific, but the experiment will give us an indication 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 experiment also helps determine if the implementations are usable in a real world setting. The bandwidth consumption will depend on the configurable options of chord and pastry. This makes a fair comparison hard. Configuration values will chosen that generally prove to give good *routing* performance.

This experiment will measure the bandwidth consumed per node over a 5 minute window per network size.

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 *routing* 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 are likely to be hosted on servers 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