Distributed Systems Project, Spring 2015

Exercise 3, Network Overlay

Jonne Airaksinen, 013932592
08.03.2015

Instructions

To run the network overlay, use the startup.sh script in the folder. To kill the node.js instances afterwards, use the kill_processes.sh script.

How the network overlay is designed

The network overlay was designed to be a 1023 node binary tree plus one additional central node. This design choice is by no means the most efficient and is also very prone to failure due to lack of alternative routes between nodes.

The design choice was purely the easiest way to tackle this problem. The size of the average and maximum routing tables remain small, four for every node except the leaf nodes, which have a routing table of two lines (two separate lines for one parent), and the central node, which has a routing table size of one. The average number of hops between nodes also remain quite low.

Calculations

Theoretical

The average hops within binary tree can be calculated from $\frac{\mathit{Internal\ Path\ Length}}{\mathit{Total\ Nodes}}$. The following formula *[Guerrilla Capacity Planning: A Tactical Approach to Planning for Highly Scalable Applications and Services, Neil J. Gunther]* calculates this number for us.

$$\frac{\sum_{k=1}^{h} k*2^{k-1}}{\sum_{k=1}^{h} 2^{k-1}} = \frac{\sum_{k=1}^{10} k*2^{k-1}}{\sum_{k=1}^{10} 2^{k-1}} = \frac{9217}{1023} \approx 9$$

Thus, the average hops needed for traveling between two nodes is 9.

From how the overlay is built it is easy to determine the minimum, maximum and average size of routing tables. As mentioned before, the central node only has *one row* in its routing table, the leaf nodes all have *one row* in their routing table and the rest of the nodes have *four rows* in their routing tables.

Thus, the average routing table size is $\frac{1*1+512*1+510*4}{1024} \approx 2{,}5$

The product of average number of hops for delivering a message, H, and the maximum routing table, R, is thus $H*R=9*4=36$.

Experimental

The experimental results reinforce the theoretical calculations, when generating the routing tables, we can see that the minimum, maximum and average routing table sizes match up to, give or take some rounding errors in the average routing table size.

Minimum routing table size: 1

Maximum routing table size: 4

Average routing table size: 2.4970703125

Implementing the network overlay using Node.js was difficult in some ways. I had problems with the delivery of messages being uncertain when too many of the nodes were messaging the overlay controller at the same time. At first the problem was only when the node routing tables were ready, since most of the nodes were ready with task at the same time.

This caused the buffer of the overlay controller to overflow, losing some of the messages. The problem was easily avoided by telling the nodes to retry until they get confirmation from the controller, just a boolean array of length 1024 was required for checking if a message from a certain node had already arrived, thus preventing double messages.

Preventing receiving double messages was not as easy with the actual message sending. To check if the message has already arrived, a 1024 x 1024 array was required. This didn't seem like such a good idea performance wise, but it solved the problem of receiving multiple messages from one properly delivered message.

Before this addition the experimental value of average hops was about 3.5, due to multiples of the short routes being calculated in the total number of hops, ruining the average in the process.

However, after my changes I ran into some problems with the message passing that I could not solve, forcing me to return to a previous version of the overlay and return that, flawed version.

Analysis of the chosen design

The overlay has only single routes between two nodes, which helps with the implementation and estimation of average hops, but just one node failing isolates the network overlay into two separate networks.

The overlay was very easy to implement, but is still quite efficient.

Comparison between experimental results and theoretical predictions

The results and predictions are very close to each other, as expected, since the structure of the network overlay is very simple, making the calculations easy. The only problem was the unexpected values of the average hops, caused by the problems with network buffer overflow when using

Node.js.