4. Übung, SS 2013
Verteilte Systeme/ Distributed Systems
Ausgegeben am 10.6.2013
Vorstellung 20.6.2013

Katinka Wolter
Benjamin Güldenring
Inst. für Informatik
Freie Universität Berlin

Assignment 1. Chord

Consider the Chord system as we discussed

1. If we insert a node into a Chord system, do we need to instantly update all the finger tables?
2. assume that node 6 has just joined the network. What would its finger table be and would there be any changes to other finger tables? Solution: For node 7 the solution is: Let us first consider the finger table for node 7. Using the same method as we introduced when discussing Chord, it can be seen that this table will be [9, 9, 11, 18, 28]. For example, entry #2 is computed as $succ(7+21) = succ(9) = 9$. More tables will need to change, however, in particular those of node 4 (which becomes [7,7,9,14,28]), node 21 ([28,28,28,1,7]) and node 1 ([4,4,7,9,18]).
3. assume a system for which k bits of an m-bit identifier space have been reserved for assigning to superpeers. If identifiers are randomly assigned, how many superpeers can one expect to have in an N-node system? Solution: The total number of superpeers in an overlay of N nodes will be equal to $\min\{2^{k-m}N, 2k\}$.

Assignment 2. Logical Time

The programmers at Flaky Computer Corporation have invented four notions of 'illogical time' for asynchronous send/receive network systems. Each of the four notions of illogical time results from dropping exactly one of the four properties for logical time. They think that the four properties might be useful for some applications. For each of their four notions

1. Describe an algorithm transformation that imposes that kind of illogical time
2. Discuss possible applications

Assignment 3. The T-Man protocol

Download the peersim library for the T-Man protocol from `http://peersim.sourceforge.net/code/T-Man.zip` and read sections 1-6.5 from the corresponding paper [1] so that you understand the basic workings of the protocol. Solution: Goals of this assignment: understand why we need to repeat experiments and how peersim helps automating this task; See that peersim is/was actually used in scientific practice and how it helps "verifying" (at least in the simulation) the results; On the way learn about the existance of "awk" and read a small bit of scientific literature on a distributed protocol (and learn a bit of t-man of course...)

We now want to verify the results from Fig. 10 from [1]:

1. What does this figure tell us? Solution: How long it takes (average) till the overlay network is fully constructed (for the given goal-topology).
2. Have a look at the configuration files `sort-rumor-scale.cfg` for Anti-Antropy Push and `sort-rumorpp-scale.cfg` for Anti-Antropy Push-Pull. What do the parameters `simulation.experiments` and range.0 do? Solution: repeating experiments and running repeating experiments for a range of values
3. Run the experiments from the above configuration files for the range 10 to 18. **HINTS:** have a look at the README from the zip file; it is OK if you set `simulation.experiments` to a much smaller value - say 3 or 4.
4. Use the output of the experiments to produce a graph similar to Fig. 10 in the paper.
5. Which configuration performs "best" in the evaluation so far? Solution: push-pull

3.1 Bonus Question:

Adjust the awk script below so that it calculates the standard deviation instead of the min and max values.

awk script for assignment 3

The output of the experiment needs further processing before we can visualize it with gnuplot. The awk (see `http://www.gnu.org/software/gawk/`) script below uses the last timestamps of converging experiment runs and calculates the min, max and average values. You can run the script using

```
$ awk -f script.awk data.txt
```

where `script.awk` is the script file and `data.txt` is the output of one set of experiments.

If you are working on an UNIX machine awk is most likely already installed. If you are working on a windows machine you can get awk from `http://gnuwin32.sourceforge.net/packages/gawk.htm`).

```
BEGIN {
  LARGE = 10^10;
}

function initialize(nr){
  min[nr] = LARGE;
  max[nr] = -1;
  avg[nr] = 0;
  navg[nr] = 0;
  initialized[kNr] = 1;
}

"control.ring" {
  kNr = $5;
  if($6 == "ENDTIME" && $9 == "NEXT")
  {
    if(initialized[kNr] == 0)
      initialize(kNr);
    time = $7;
    avg[kNr] += time;
    navg[kNr]++;
```

```
    if(min[kNr] > time)
      min[kNr] = time;
    if(max[kNr] < time)
      max[kNr] = time;
  }
}

END {
  for(i in initialized)
    printf("%d %f %f %f\n", i, avg[i]/navg[i], min[i], max[i]);
}
```

Hint for gnuplot

To produce error bars (with the min and max values calculated above) you have to use a gnuplot command like

```
gnuplot> plot "data.txt" w yerrorbars, "" w lines
```

assuming the output of the awk script is in the file `data.txt`.

Assignment 3. References

[1] Alberto Montresor and Ozalp Babaoglu, *T-Man: Gossipbased fast overlay topology construction. Computer Networks*, 2009, (Online at `http://disi.unitn.it/~montreso/pubs/papers/comnet09.pdf`)