Distributed Systems - Assignment 4

Markus Roth

November 15, 2016

1 Exercise 1: Synchronization

I assume the indexing and search engine works in the following manner: A large number of machines work together. Each machine uses multiple threads to visit and download websites. The downloaded website is processed by stripping out irrelevant information, then added to various indexes. These indexes are also distributed across many different machines. Search queries are handled by a load balancer distributin them to many machines. Each of the machines then parses the query, decides what index to use, and forwards the query to one or more machines that serve the indexes.

There are therefore nodes for: Spidering, preprocessing and indexing, hosting indexes and content, load balancing and query forwarding / result merging.

Websites can and do often change, so the time of retrieval is important. Multiple spiders might read the same website only a sshort time after each other, so their times need to be synchronized so the index can always show the most recent copy.

The spiders also need to share state about what sites they already visited and at what time, to schedule re-visiting of websites at certain intervals.

So all the nodes need to synchronize their wall-clock times, and need to have some synchronized state.

2 Exercise 2: Vector Clocks

2.1 Part 1: Lamport Clock

The rules are:

```
e is a local event or a send event:
If e has no local predecessor, L(e) = 1 else L(e) = L(e') + 1

e is a receive event, s the corresponding send event:
If e has no local predecessor, L(e) = L(s) +1
If eâĂŹ is the local predecessor of e, L(e) = max { L(s), L(e') } + 1
```

Therefore the clocks states are:

```
b: 1 (send event, no local predecessor)
f: 1 (send event, no local predecessor)
```

```
a: 2 (receive event, no local predecessor, send event f = 1)
c: 2 (local event, local predecessor b = 1)
e: 3 (send event, local predecessor c = 2)
g: 2 (send event, local predecessor f = 1))
d: 3 (receive event, local predesessor a = 2, sender b = 1)
j: 4 (local event, local predecessor e = 3)
h: 4 (receive event, local predecessor g = 2, sender e = 3)
i: 5 (receive event, local predecessor j = 4, sender g = 2)
```

2.2 Part 2: Vector Clock

```
b: (0, 1, 0)
f: (0, 0, 1)
a: (1, 0, 1)
c: (0, 2, 0)
e: (0, 3, 0)
g: (0, 0, 2)
d: (2, 1, 1)
j: (0, 4, 0)
h: (0, 3, 3)
i: (0, 5, 2)
```

2.3 Part 3: Happened-Before and Concurrency

a) A and I happen concurrently, it is impossible to determine which one happens first as there is no interaction.

b) F happens before I, because G happens after F and I happens after G. In the Lampart Clock numbering, F=1 and I=5, this is enough to decide that F happened after I. In Vector Clocks numbering, F=(0,0,1) and I=(0,5,2). I is clearly higher than F because each number in the vector is higher.

c) H happens after B, because C happens after B, E happens after C, and H happens after E. In the Lampart Clock numbering, H=4 and B=1, so H is after B (they are causally dependent). In Vector Clocks, H=(0,3,3) and B=(0,1,0). Here, H is clearly after B because all numbers in the vector are higher.