TDT4225 - Exercise 1

Håkon Ødegård Løvdal

September 2015

Problem 1. Peer-to-peer networks

a)

Structured peer-to-peer networks are structured with an specific topology (global structure) to ensure that each resource can easily be found/located. To locate a resource one must do a lookup on key (the GUID), which normally is a hash of the IP-address and/or more. This lookup is done in the routing overlay, which routes the request to the node containing the resource. A resource can be an object, or a part of an object. By using this targeted lookup, one can guarantee a lookup in $O(log\, n)$.

In unstructured peer-to-peer networks there are no specific topology/global structure. Each node can connect to any other node, and it does not now if another node know more. This means that there are no structure in the routing overlay. Due to this, the local networks can optimize interally. This poor structure means that one must search througout the entire network to find all nodes holding a certain resource.

b)

p =	GUID prefixes and corresponding nodes															
0	0	1	2 241	3 3EB	4	5	6	7 7CB	8	9	A	B B23	C C6C	D	E EE0	F FF2
1	50	51	52	53	54 540	55	56	57	58	59	5A	5B	5C	5D	5E 5EC	5F
2	590	591	592 592	593	594	595	596	597 597	598	599	59A	59B	59C	59D	59E	59F

Table 1: Prefix routing table for 599

p =	GUID prefixes and corresponding nodes															
0	0	1	2 241	3 3EB	4	5 5EC	6	7 7CB	8	9	A	B B23	C	D	E EE0	F FF2
1	C0	C1	C2	C3	C4	C5	C6	C7	C8	C9	CA	CB	CC	CD	CE	CF
2	C60	C61	C62	C63	C64	C65 C65	C66	C67	C68	C69	C6A C6A	C6B	C6C C6C	C6D	C6E	C6F

Table 2: Prefix routing table for C61

c)

Alternative 1: FF2

Initial values: $A = C61, D = FC7, L = 2 \times l = [B23, C65]$ and $l = 2$

Line 1 in Pastry's routing algorithm will fail since $(L_l < D < L_l)$ returns false. Therefore the algorithm tries to match against the routing table. Here $p = 0$ and $i = 1$. Therefore A will send M to $R[0, 1]$ which is FF2. FF2 will have FC7 in its leaf set [FC7, 219] and since the algorithm will forward M to L_i where L_i is the destination, FC7 will be reached.

Alternative 2: F84

Initial values: $A = C61, D = FC7, L = 2 \times l = [B23, C65]$ and $l = 2$

In this alternative, F84 is choosen as corresponding node for prefix F in C61's routing table. Just like in alternative 1, line 1 in Pastry's routing algorithm will fail since $(L_l < D < L_l)$ returns false. Therefore the algorithm tries to match against the routing table. Here $p = 0$ and $i = F$. Therefore A will send M to $R[0, F]$ which is F84. Just like alternative 1, F84 will have FC7 in its leaf set [F19, FC7]. Also here, since the algorithm will forward M to L_i where L_i is the destination, FC7 will be reached.

Alternative 3: F19

Initial values: $A = C61, D = FC7, L = 2 \times l = [B23, C65]$ and $l = 2$

In this alternative, F19 is choosen as corresponding node for prefix F in C61's routing table. Just like in alternative 1 and 2, line 1 in Pastry's routing algorithm will fail since $(L_l < D < L_l)$ returns false. Therefore the algorithm tries to match against the routing table. Here $p = 0$ and $i = F$. Therefore A will send M to $R[0, F]$ which is F19.

Updated values: $A = F19, D = FC7, L = 2 \times l = [EE0, F84]$ and $l = 2$

Now $p = 1$ and $i = C$. Since R[1, C] = FC7 the algorithm will forward M to FC7, and FC7 will ble reached

Problem 2 - Time

a)

Given the two events e and f, where the logical clock values L are such that $L(e) < L(f)$ we can not deduce the "happend-before-releation" $e \rightarrow f$.

This is because while using the Lamport clock no process knows anything about the other processes. If you know $e \rightarrow f$, then you can deduce $L(e) < L(f)$. By looking at the Lamport timestamps alone, we can not deduce the "happend-before-relation". The value of $L(e)$ for instance is only a representation of the timestamp in the process it occured and is implemented as a simple incremental counter. In an event were a message is sent from one process to another, the timestamp is set to: $L_i = max(L_i, t) + 1$. Therefore it does not give any information on wheter the event occured before, concurrent or later than for instance $L(f)$.

If we use vector clocks each process implement a vector of size N, where N is the number of processes. Each process keeps track of its local timestamp, but on a message send they piggyback their vector. By doing this every process will know other process' states on receiving a message. This removes the issue in deducing $e \rightarrow f$ when $L(e) < L(f)$.

b)

b. (4,0,0)

k. (4,2,0)

m. (4,3,0)

c. (4,3,2)

u. (4,4,0)

n. (5,4,0)

Problem 3 - Global state

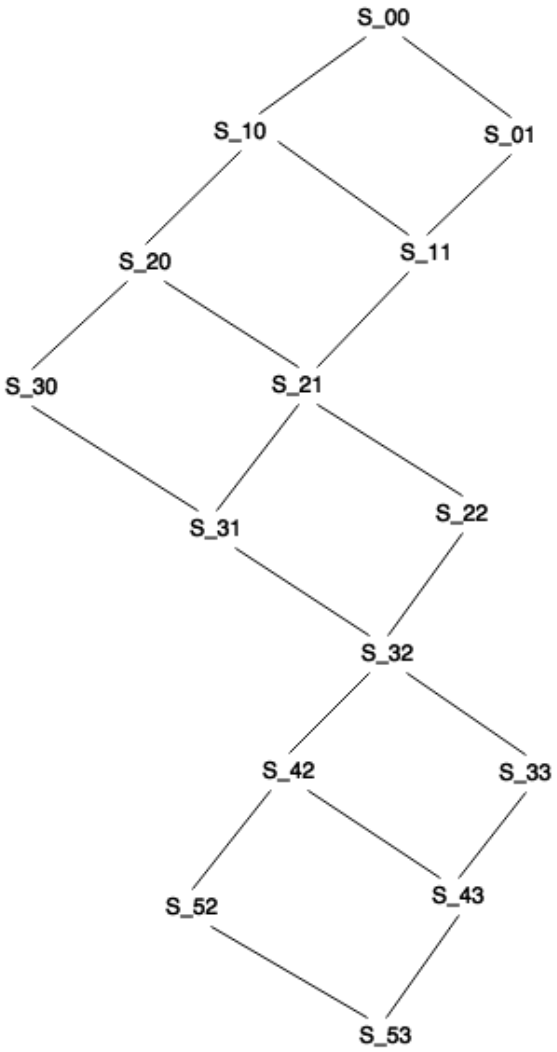

Figure 1: All possible consistent states the processes may have had