27/33

Verteilte Systeme I
Winter Term 2019/20

G2T1 – Assignment 1 (theoretical part)

Felix Bühler 2973410 | Clemens Lieb 3130838 | Steffen Wonner 2862123 | Fabian Bühler 2953320

3. November 2019

10 Transparency Levels

a) 2/2

The access is location transparent, because the location is hidden behind an unresolved URL. The actual address of the webservice is automatically resolved through DNS acting as service discovery mechanism. Also the actual song is referenced by title and not by a specific location.

b) 1/2

The service is not replication transparent because there is a unique name for every location that has to be checked separately. The client has to know all available replications for this to work.

c) 2/2

The access is replication transparent, because it is not apparent whether Otto checks one or multiple web services. It's furthermore unknown which of the two replication locations did provide the file in the end.

d)

i. It is only possible to write to all copies at once so all copies will always have the same value. This implies that independently from when or which copy is read the answer is always up to date. 2/2

ii. Write operations are performed on $n-1$ copies, as required by the locking constraints (only locks for these $n-1$ copies are acquired). A write operation also marks the time when the value was updated. Read operations are performed on 2 different randomly selected copies, the client accepts the "newer"version according to the timestamps of the updates. The reading operation only needs to lock the two selected copies. 3/3

Felix Bühler
Clemens Lieb
Steffen Wonner
Fabian Bühler

Verteilte Systeme I
Winter Term 2019/20 G2T1 – Assignment 1 (theoretical part)

5.5/10

2 System Models

a) 4/4

- There are no node failures — no process failures
- No message is lost — no communication failures
- The maximum possible delay has a known upper bound
- There is a limited amount of clock skew between the processes

b)

i. Assuming an upper limit s of the clock skew between the processes, the process P_i sends a broadcast message $2 \cdot s \cdot \delta t + \delta t$ time units after receiving $i-1$ messages. The first process sends immediately.

   This allows for instant delivery to a process skewed by the maximum amount of time "forwards" while keeping the order of messages intact when sending the message even with the maximum delay δt to a process skewed by the maximum amount of time "backwards". 0/4

ii. In the worst case, the synchronization protocol takes $n \cdot \delta t + (n-1) * (2 \cdot s \cdot \delta t + \delta t)$ time units. The first part $n \cdot \delta t$ is the maximum possible network delay. The second part $(n-1) * (2 \cdot s \cdot \delta t + \delta t)$ is the waiting time of all processes except the first one that does not have to wait. Clock skew can be ignored here as we do not compare the timestamps of the first send with the last receive. 1.5/2

b i) Delta t is not fixed, it can be individual for each receiver.
Each process receives the message in a time between d and k.

b ii) The worst case network delay until a message is received is k.
The maximum time a process needs to wait (from 2b i)) is k-d.
If you consider this your formula would be correct.

Felix Bühler
Clemens Lieb
Steffen Wonner
Fabian Bühler

5.5/6

3 Three-Army-Problem

a)

To enable a resolution the failure model should exclude both crashing and byzantine failures. General Omissions, so long as their occurrences are bounded, can be resolved by retrying communication sufficiently often. Message delivery also has to have an upper bound on the delay.

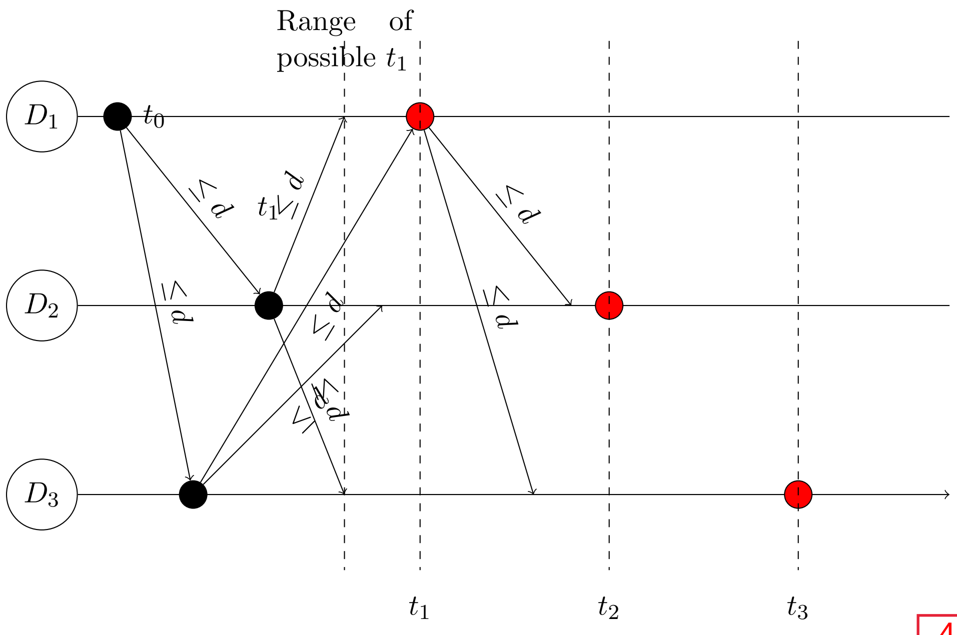
1.5/2 No failures on the communication channel
No process failure

b)

Phase 1: Determining order of attack The initiating division sends a messenger with it's own size and the time t_0 to both other divisions. A division receiving a first message with a division size sends it's division size to the two other divisions. If it has received two messages, it calculates the order of attack. If it is the largest division it initiates Phase 2.

Phase 2: Attack coordination The largest division receives the second message at $t_1 = t_0 + r, r \leq 2d$. The largest division attacks at t_1 and sends a message with that time to both other divisions. The new message (and any outstanding messages from the first phase) will arrive before the other divisions need to attack as $k > d$. The second largest division attacks at time $t_2 = t_1 + k$, the smallest division attacks at $t_3 = t_1 + 2k = t_2 + k$.

Since all clocks are perfectly synchronized we only need to consider message delivery times. t_1 is minimal, because it's the first moment when the largest division knows it is the largest. Since all divisions must have sent their size *before* t_1 – otherwise the information would not be available – all divisions know their position in the order of attack at $t_1 + d < t_1 + k$, which implies that $t_2 > t_1 + d$ as required to allow the second division to attack at the specified time. This holds especially for $t_3 > t_2$. And because all attacks must be at least k apart, $2k$ is the minimal span of time between the first and last attack.



4/4

Felix Bühler
Clemens Lieb
Steffen Wonner
Fabian Bühler

6/6

4 System Availability

a) $A_x = \frac{80t}{100t} = 80\%$ 1/1

$A_y = \frac{60t}{100t} = 60\%$

b) Overall statistical availability:

$A_S = 1 - P(Y = \text{down} \cup X = \text{down}) = 1 - ((1 - A_x) \cdot (1 - A_y)) = 1 - (0.2 \cdot 0.4) = 1 - 0.08 = 0.92 = 92\%$ 1/1

Availability in the concrete example:

$A_S = \frac{80t}{100t} = 80\%$

c) $P(Y = \text{up}|X = \text{up}) = \frac{P(Y=\text{up}\cap X=\text{up})}{P(X=\text{up})} = \frac{0.6}{0.8} = 75\%$

$P(Y = \text{up}|X = \text{down}) = \frac{P(Y=\text{up}\cap X=\text{down})}{P(X=\text{down})} = \frac{0}{0.2} = 0\%$

Node Y seems to depend on Node X to be available. 2/2

d) The availability of the system (A_S) depends on the node X.

$A_S = P(Y = \text{up} \cup X = \text{up}) = P(X = \text{up}) = A_X = 0.8$

This is different from the result in b) ($A_S = 0.92$) where the assumption was that the nodes have independent failures. If the nodes are dependent then the overall system can not achieve a higher availability than any one of the nodes. 2/2