Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Systems@ETH Zürich

Prof. T. Roscoe, Prof. R. Wattenhofer

Computer Systems

— Solution to Assignment 10 —

1 Quorum Systems

1.1 The Resilience of a Quorum System

a) No such quorum system exists. According to the definition of a quorum system, every two quorums of a quorum system intersect, so at least one server is part of both quorums. The fact that all servers of a particular quorum fail implies that in each other quorum at least one server fails, namely the one which lies in the intersection. Therefore, it is not possible to achieve a quorum anymore and the quorum system does not work anymore.

b) Just 1—as soon as 2 servers fail, no quorum survives.

c) Imagine a quorum system in which all quorums overlap exactly in one single node; i.e. each element of the powerset of the remaining $n-1$ nodes joined with this special node is a quorum. This gives 2^{n-1} quorums.
Can there be more? No! Consider a set from the powerset of n servers. Its complement cannot be a quorum as well, as they do not overlap. So, from each such couple, at most one set can be part of the quorum system. This gives an upper bound of $2^n/2 = 2^{n-1}$ quorums.

1.2 A Quorum System

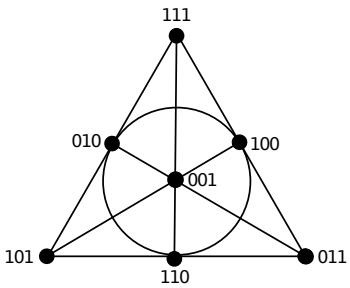

Figure 1: Quorum System

a) This quorum system consists of 7 quorums. As work is defined as the minimum expected number of servers in an accessed quorum (over all access strategies), this system's work is 3 (all strategies induce the same work on a system where all quorums are the same size). Observe that all nodes are in precisely 3 quorums, so the uniform access strategy induces the same load on all nodes. Since the quorum system is also 3-uniform, by exercise 3 it follows that the uniform strategy is optimal; it's load being 3/7.

b) The resilience is $R(\mathcal{S}) = 2$. Proof: every node is in exactly 3 quorums, so 2 nodes can be contained in at most $2 \cdot 3 = 6 < 7 = |\mathcal{S}|$ quorums, thus, if no more than 2 nodes fail, there will be at least 1 quorum without a faulty node. If, on the other hand, for example, the nodes 101, 010 and 111 fail, no quorum can be achieved; see also exercise 1a).

1.3 S-Uniform Quorum Systems

Definitions:

s-uniform: A quorum system $\mathcal{S}$ is *s-uniform* if every quorum in $\mathcal{S}$ has exactly s elements.
Balanced access strategy: An access strategy Z for a quorum system $\mathcal{S}$ is *balanced* if it satisfies $L_Z(v_i) = L$ for all $v_i \in V$, for some value L.

Claim: An s-uniform quorum system $\mathcal{S}$ reaches an optimal load with a balanced access strategy, if such a strategy exists.

a) In an s-uniform quorum system each quorum has exactly s elements, so independently of which quorum is accessed, s servers have to work. Summed up over all servers we reach a total load of s, which is the work of the quorum system. As the load induced by an access strategy is defined as the maximum load on any server, the best strategy would be to evenly distribute this work on all servers. If such a strategy exists, then it is therefore optimal.

b) Let $V = \{v_1, v_2, ..., v_n\}$ be the set of servers and $\mathcal{S} = \{Q_1, Q_2, ..., Q_m\}$ an s-uniform quorum system on V. Let Z be an access strategy, thus it holds that: $\sum_{Q \in \mathcal{S}} P_Z(Q) = 1$. Furthermore, let $L_Z(v_i) = \sum_{Q \in \mathcal{S}; v_i \in Q} P_Z(Q)$ be the load of server v_i induced by Z.

Then it holds that:

$$\begin{aligned}\sum_{v_i \in V} L_Z(v_i) &= \sum_{v_i \in V} \sum_{Q \in \mathcal{S}; v_i \in Q} P_Z(Q) = \sum_{Q \in \mathcal{S}} \sum_{v_i \in Q} P_Z(Q) \\ &= \sum_{Q \in \mathcal{S}} P_Z(Q) \cdot |Q| \overset{*}{=} \sum_{Q \in \mathcal{S}} P_Z(Q) \cdot s = s \cdot \sum_{Q \in \mathcal{S}} P_Z(Q) = s\end{aligned}$$

The transformation marked with an asterisk uses the uniformity of the quorum system.

To minimize the maximal load on any server, the optimal strategy would be to evenly distribute this load on all servers. Thus, if a balanced access strategy exists, this leads to an optimal system load of s/n.

Note: A balanced access strategy does not always exist for example for the following 2-uniform quorum system: $V = \{1, 2, 3\}$, $\mathcal{S} = \{\{1, 2\}, \{1, 3\}\}$. We have $\min\{L_Z(2), L_Z(3)\} < L_Z(1) = 1$ for any access strategy on this system.

2 Distributed Storage

Quiz

2.1 Hypercubic Networks

The hypergraphs are drawn as shown in Figure 2. The 4-dimensional hypercube $M(2, 4)$ can also be drawn as two cubes side-by-side with connected corners.

Basic

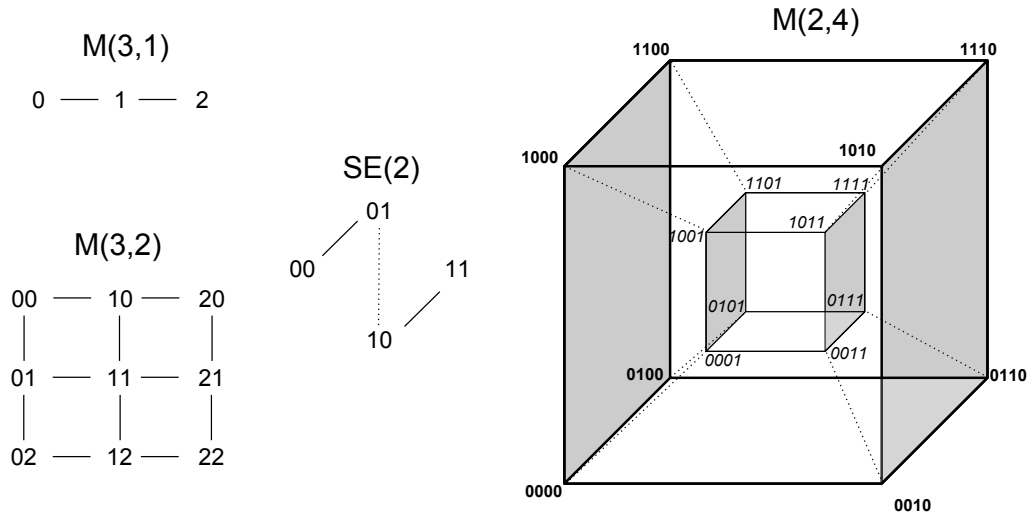

Figure 2: Drawings of $M(3,1)$, $M(3,2)$, $SE(2)$ and $M(2,4)$. Note that we have used dashed lines and solid lines for $SE(2)$ the other way around compared to the lecture.

2.2 Iterative vs. Recursive Lookup

a) In the recursive lookup there is no difference between a request originating at the node and a request being forwarded. Only once the lookup is finished do we need to care about forwarding the result to the previous node or returning it to the caller. This allows the same lookup logic to be reused for both. Furthermore, if the response is returned through the same path as the request was sent through, then the intermediate nodes can cache the result, potentially speeding up future lookups and distributing the load of a popular item.

b) Recursive lookups can easily be misused to mount Denial-of-Service attacks, since a single request message from an originating node is forwarded over multiple hops. Each hop multiplies the impact this message has on the network. Thus the attacker's bandwidth is potentially multiplied by the number of hops the request is routed through. Furthermore, if the result is returned over the same path as the request, then the attacker is hidden behind a number of hops and the victim only sees traffic originating from the last hop.

2.3 Building a set of Hash functions

The salted hashing function derivation allows random access to any of the derived hashing functions without having to recompute the intermediate functions, as is the case in the iterative hashing function derivation scheme. Furthermore, iterative hashing allows a user which knows the first hash value to deduce all the others by repeatedly applying h. This information can be used by an attacker to target specific nodes storing a movie.

Advanced

2.4 Multiple Skiplists

a) Figure 3 shows the structure of a multi-skiplist with 8 nodes and 3 levels. Notice that each of the lists would wrap around at the ends.

b) Unlike in the single skiplist each node now has a constant degree of $2 \cdot d$, i.e., on each level it has a right and a left neighbor, including the full list at $l = 0$.

c) The number of hops is still $O(\log n)$, just like for the simple skiplist.

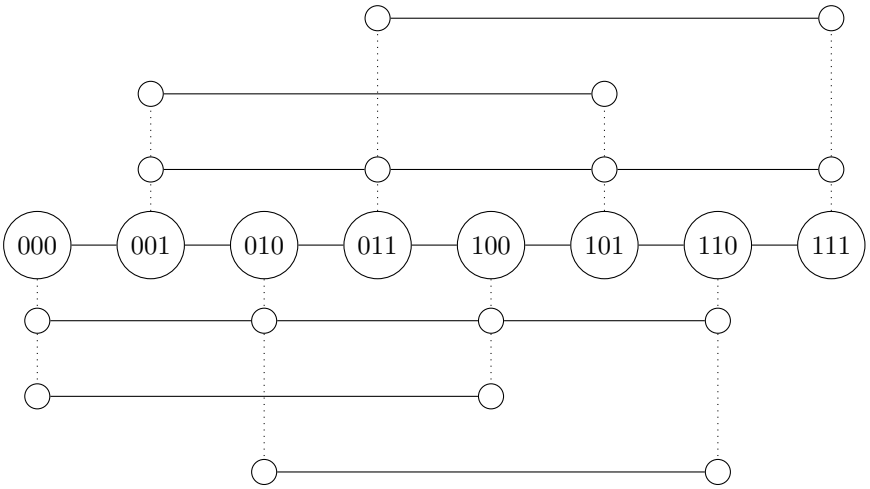

Figure 3: A multi-skiplist with 3 levels and 8 nodes.