Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Systems@ETH Zürich

Prof. T. Roscoe, Prof. R. Wattenhofer

Computer Systems

Assignment 9

1 Quorum Systems

Quiz

1.1 The Resilience of a Quorum System

a) Does a quorum system exist, which can tolerate that all nodes of a specific quorum fail? Give an example or prove its nonexistence.

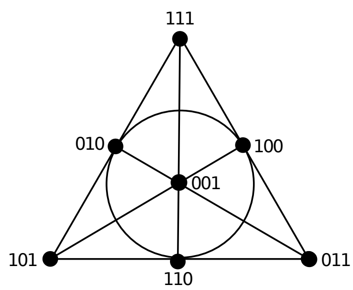
b) Consider the *nearly all* quorum system, which is made up of n different quorums, each containing $n-1$ servers. What is the resilience of this quorum system?

c) Can you think of a quorum system that contains as many quorums as possible?
Note: the quorum system does not have to be minimal.

Basic

1.2 A Quorum System

Consider a quorum system with 7 nodes numbered from 001 to 111, in which each three nodes fulfilling $x \oplus y = z$ constitute a quorum. In the following picture this quorum system is represented: All nodes on a line (such as 111, 010, 101) and the nodes on the circle (010, 100, 110) form a quorum.



a) Of how many different quorums does this system consist of and what are its work and its load?

b) Calculate its resilience f. Give an example where this quorum system does not work anymore with $f+1$ faulty nodes.

Advanced

1.3 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) Describe in your own words why this claim is true.

b) Prove the optimality of a balanced access strategy on an s-uniform quorum system.

2 Approximate Agreement

Quiz

2.1 Asynchronous Protocols in Synchronous Networks

In the lecture, you have seen a Single-Value Reliable Broadcast algorithm (Algorithm 20.11). Sometimes, ideas used in the asynchronous model also lead to cute properties in the synchronous model. Let us analyze the algorithm below in a **synchronous** network where $f < n/3$ of the nodes are byzantine.

Algorithm 1 Single-Valued Reliable Broadcast, But in a Synchronous Network

Code for sender v_S with input x_S:
Round 1: Send `msg`(x_S) to everyone.

Code for node v:
Round 2:
    If you received a message `msg`(x) from the sender:
    Send `echo`(x) to everyone.

Round 3 or later:
    Upon receiving `echo`(x) from $n - f$ distinct nodes or
            `ready`(x) from $f + 1$ distinct nodes:
        Send `ready`(x) to everyone.

Round 4 or later:
    Upon receiving `ready`(x) from $2f + 1$ distinct nodes:
    Accept `msg`(x).

a) What strategy should the byzantine nodes use so that two correct nodes accept different values?

b) Assume that a correct node v has accepted `msg`(x). Explain why every correct node accepts `msg`(x) within two additional communication rounds.

c) Assume that a correct node v has not accepted a value by the end of round 4. What does that tell v about the sender v_S?

Basic

2.2 From Approximate Agreement to Byzantine Agreement

We want to design an **asynchronous** byzantine agreement algorithm (where nodes' inputs are bits) that relies on Algorithm 20.22 from the lecture nodes. Recall that Algorithm 20.22 achieves asynchronous approximate agreement even when $f < n/3$ of the nodes are byzantine.

Nodes proceed as follows: every node joins Algorithm 20.22 with its input bit as initial value. Once a node obtains a value x from Algorithm 20.22, it outputs 0 if $x < 0.5$ and 1 otherwise.

a) Does all-same validity hold?

b) What about agreement?

c) Assume an ideal shared coin that enables the nodes to agree on a uniformly distributed random value in $(0, 1)$. Once $f + 1$ nodes query this shared coin, the random value is sampled and all nodes learn it eventually.

How can we use this coin to achieve agreement except with probability 10^{-2023}?

Advanced

2.3 Unbounded Input Space: Quick Fix

The approximate agreement algorithms presented in the lecture rely on a publicly known `max_range` that the input space should satisfy. This allows us to (overestimate) a sufficient number of iterations. To drop this assumption **in the synchronous model** (Algorithm 20.10), we will build a mechanism that enables each node to (over)estimate a `max_range` based on the nodes' inputs. Hence, if X denotes the multiset of correct inputs, we will ask each node to estimate $\max X - \min X$.

a) How would obtaining agreement on $\max X - \min X$ help?

b) Describe in your own words why correct nodes cannot agree on $\max X - \min X$.

Instead, each node will try to *estimate* the initial range X. This can be done using one round of communication preceding the for loop of Algorithm 20.10.

c) Write an algorithm that uses one round of communication and allows each correct node v to obtain an estimation $\texttt{max\_range}_v \geq \max X - \min X$.

d) How can the algorithm from Task **c)** be used to replace the hard-coded value I in Algorithm 20.10? Keep in mind that nodes do not obtain the same value $\texttt{max\_range}_v$.

e) Can you provide an upper bound on the number of iterations in your solution in Task **d)**?