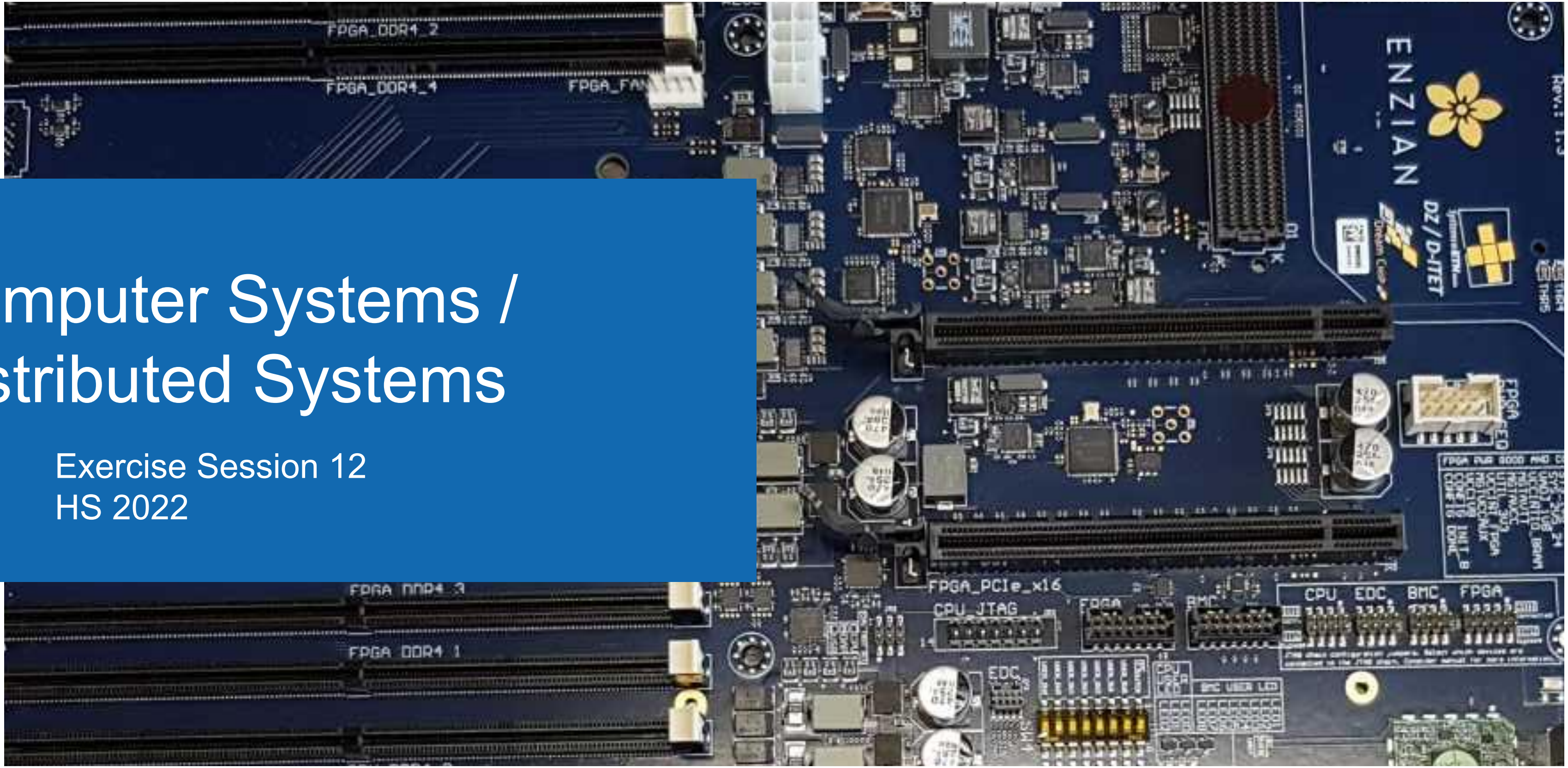
Slides last updated:
25.11.2022

Computer Systems / Distributed Systems

Exercise Session 12
HS 2022



Game Theory

Prisoner's Dilemma - matrix representation of games

	u / v	Player u	
		Cooperate	Defect
Player v	Cooperate	1 1	0 3
	Defect	3 0	2 2

Game Theory - Terminology

Strategy	move
Strategy profile	set of strategies for all players specifying all actions in a game
Social optimum (SO)	
Dominant strategy (DS)	
Dominant strategy profile	
Nash equilibrium (NE)	

Example: Prisoners Dilemma

u / v		Player u: Cooperate	Player u: Defect
Player v	Cooperate	1 (u), 1 (v)	0 (u), 3 (v)
	Defect	3 (u), 0 (v)	2 (u), 2 (v)

Strategy: Player v will play “Cooperate”

Strategy profile: Player v will play “Cooperate” and player u will play “Defect”

Dominant Strategy:

Social optimum:

Nash equilibrium:

Game Theory - Terminology

Strategy	move
Strategy profile	set of strategies for all players specifying all actions in a game
Social optimum (SO)	Strategy profile with the best sum of outcomes over players
Dominant strategy (DS)	The move that’s never worse than another strategy for a player
Dominant strategy profile	Every player plays a dominant strategy
Nash equilibrium (NE)	

Example: Prisoners Dilemma

u / v		Player u	
		Cooperate	Defect
Player v	Cooperate	1 1	0 3
	Defect	3 0	2 2

Strategy: Player v will play “Cooperate”

Strategy profile: Player v will play “Cooperate” and player u will play “Defect”

Dominant Strategy: Defect (if other player cooperates: 0<1; if other player defects 2<3)

Social optimum: Cooperate-Cooperate (cost: 2)

Nash equilibrium:

Game Theory - Terminology

Strategy	move
Strategy profile	set of strategies for all players specifying all actions in a game
Social optimum (SO)	Strategy profile with the best sum of outcomes over players
Dominant strategy (DS)	The move that's never worse than another strategy for a player
Dominant strategy profile	Every player plays a dominant strategy
Nash equilibrium (NE)	Strategy profile such that nobody can improve by unilaterally changing their move

Example: Prisoners Dilemma

	u v	Player u	
		Cooperate	Defect
Player v	Cooperate	1 1	0 3
	Defect	3 0	2 2

Strategy: Player v will play “Cooperate”

Strategy profile: Player v will play “Cooperate” and player u will play “Defect”

Dominant Strategy: Defect (if other player cooperates: 0<1; if other player defects 2<3)

Social optimum: Cooperate-Cooperate (cost: 2)

Nash equilibrium: Defect-Defect (cost: 4)

Selfish Caching

Consider a network. Nodes can either cache a file or fetch it through the network from another node. At least one node should store the file.

As a game:

- **Strategy:** cache or not cache
- **Cost:** 1 if cache, otherwise (shortest path to cache) * demand
  (Note: path lengths are symmetric (if undirected) but demands might vary)

Selfish Caching - Algorithm

Systems@ETH zürich

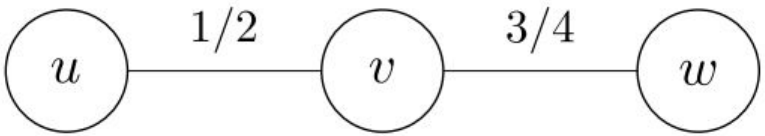
Distributed Computing

Algorithm 25.7 Nash Equilibrium for Selfish Caching

$S = \{\}$ //set of nodes that cache the file
repeat
  Let v be a node with maximum demand d_v in set V
  $S = S \cup \{v\}, V = V \setminus \{v\}$
  Remove every node u from V with $c_{u \leftarrow v} \leq 1$ ← remove all candidates that are better off by fetching
until V = {}

$c_{u \leftarrow v}$ = cost for u of fetching from v, i.e. u-v-path length * demand of u

Selfish Caching - Example



With demands all 1

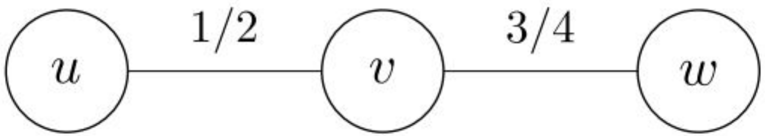
There are 2 NE, both can be found with algorithm depending on the start node:

Optimistic **NE** (start algo at v): **?**

Pessimistic **NE** (start algo at u or w): **?**

Social Optimum: **?**

Selfish Caching - Example



With demands all 1

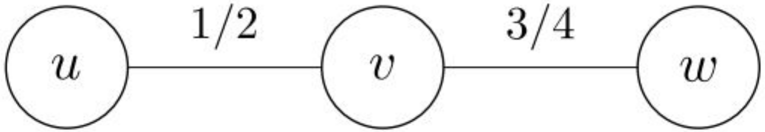
There are 2 NE, both can be found with algorithm depending on the start node:

Optimistic **NE** (start algo at v): **v caches** ⇒ Cost = 1/2 + 1 + 3/4 = **9/4**

Pessimistic **NE** (start algo at u or w): **?**

Social Optimum: **?**

Selfish Caching - Example



With demands all 1

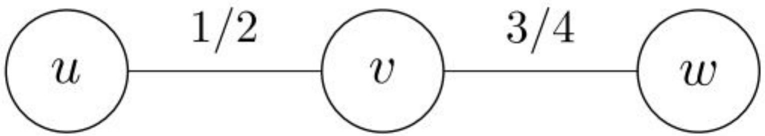
There are 2 NE, both can be found with algorithm depending on the start node:

Optimistic **NE** (start algo at v): **v caches** ⇒ Cost = 1/2 + 1 + 3/4 = **9/4**

Pessimistic **NE** (start algo at u or w): **u & w cache** ⇒ Cost = 1 + 1/2 + 1 = **10/4**

Social Optimum: **?**

Selfish Caching - Example



With demands all 1

There are 2 NE, both can be found with algorithm depending on the start node:

Optimistic **NE** (start algo at v): **v caches** ⇒ Cost = 1/2 + 1 + 3/4 = **9/4**

Pessimistic **NE** (start algo at u or w): **u & w cache** ⇒ Cost = 1 + 1/2 + 1 = **10/4**

Social Optimum: **v caches** (same as Optimistic NE) ⇒ Cost = **9/4**

Price of Anarchy

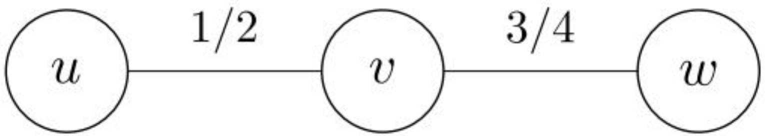
Systems@ETH zürich

Distributed Computing

Idea: With some rules, we could always enforce the social optimum. But what is the cost of having no rules (anarchy)?

- **Optimistic approach:** players will converge to “best” nash equilibrium.
  - Then, price of anarchy: $OPoA = \frac{\text{cost}(NE_+)}{\text{cost}(SO)}$
- **Pessimistic approach:** players will converge to “worst” nash equilibrium
  - Then, price of anarchy: $PoA = \frac{\text{cost}(NE_-)}{\text{cost}(SO)}$

Selfish Caching - Example



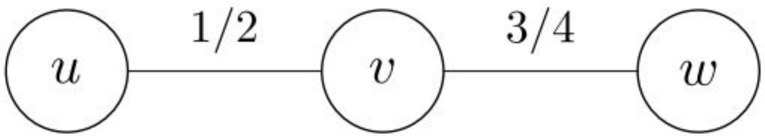
With demands all 1

Optimistic **NE**: **9/4** Pessimistic **NE**: **10/4** **Social Optimum**: **9/4**

PoA: ?

OPoA: ?

Selfish Caching - Example



With demands all 1

Optimistic **NE**: **9/4** Pessimistic **NE**: **10/4** **Social Optimum**: **9/4**

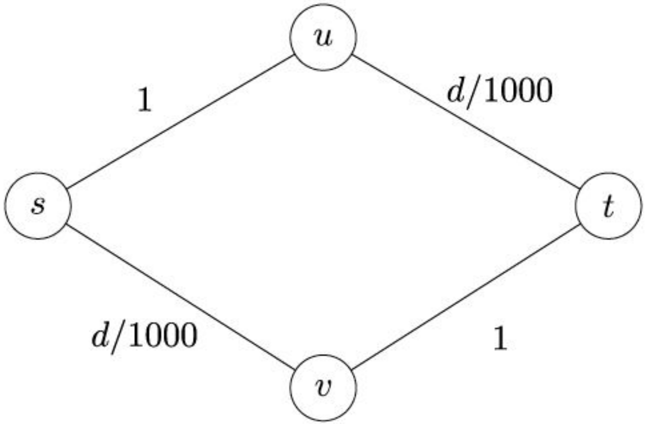
PoA: (10/4) / (9/4) = **10/9** > 1

OPoA: (9/4) / (9/4) = **1**

Braess Paradox

d = #drivers on link

NE for 1000 drivers:
split evenly across
s→u→t and s→v→t
⇒ cost = 1.5

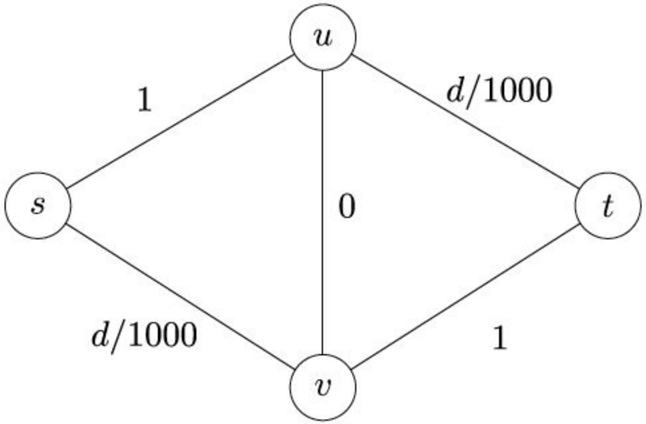


(a) The road network without the shortcut

Braess Paradox

adding link {u,v} makes the NE worse

consider even split, but then s→v→u→t costs just 1, so drivers will start switching until all choose that path ⇒ cost = 2



(b) The road network with the shortcut

Mixed Nash Equilibrium

Definition 25.16 (Mixed Nash Equilibrium). *A Mixed Nash Equilibrium (MNE) is a strategy profile in which at least one player is playing a randomized strategy (choose strategy profiles according to probabilities), and no player can improve their expected payoff by unilaterally changing their (randomized) strategy.*

Theorem 25.17. *Every game has a mixed Nash Equilibrium.*

u v		Player u		
		Rock	Paper	Scissors
Player v	Rock	0 0	1 -1	-1 1
	Paper	-1 1	0 0	1 -1
	Scissors	1 -1	-1 1	0 0

Table 23.15: Rock-Paper-Scissors as a matrix.

MNE for rock paper scissors:
Both players choose a strategy with ⅓ probability (due to symmetry)

Quiz (Assignment 11)

Systems@ETH zürich

Distributed Computing

1.1 Selling a Franc

Form groups of two to three people. Every member of the group is a bidder in an auction for one (imaginary) franc. The franc is allocated to the highest bidder (for his/her last bid). Bids must be a multiple of CHF 0.05. This auction has a crux. Every bidder has to pay the amount of money he/she bid (last bid) – it does not matter if he/she gets the franc. Play the game!

a) Where did it all go wrong?

b) What could the bidders have done differently?

Quorum Systems

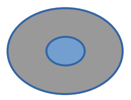
Quorum Systems

High-level functionality:

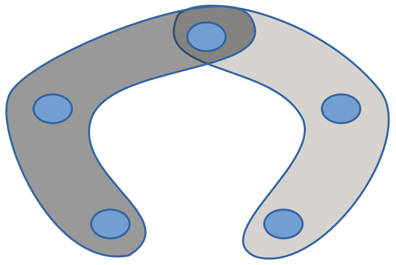
1. Client selects a free quorum
2. Locks all nodes of the quorum
3. Client releases all locks

ETH zürich | Systems@ETH zürich | Distributed Computing

Singleton and Majority Quorum Systems



Singleton quorum system



Majority quorum system
(all sets of n / 2 + 1 nodes)

Load and Work

An access strategy Z defines the probability $P_Z(Q)$ of accessing a quorum Q ∈ S such that:

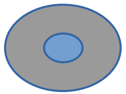
$$\sum_{Q \in S} P_Z(Q) = 1$$

Load and Work

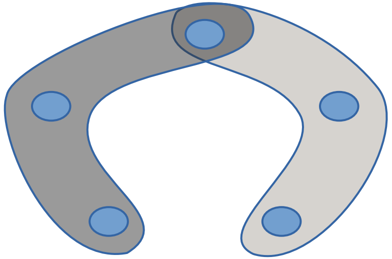
- **Load** of access strategy Z on a node v_i $$L_Z(v_i) = \sum_{Q \in S; v_i \in Q} P_Z(Q)$$
- **Load** induced by Z on quorum system S $$L_Z(S) = \max_{v_i \in S} L_Z(v_i)$$
- **Load** of quorum system S $$L(S) = \min_Z L_Z(S)$$

- **Work** of quorum Q $$W(Q) = |Q|$$
- **Work** induced by Z on quorum system S $$W_Z(S) = \sum_{Q \in S} P_Z(Q) \cdot W(Q)$$
- **Work** of quorum system S $$W(S) = \min_Z W_Z(S)$$

Load and Work



Singleton quorum system

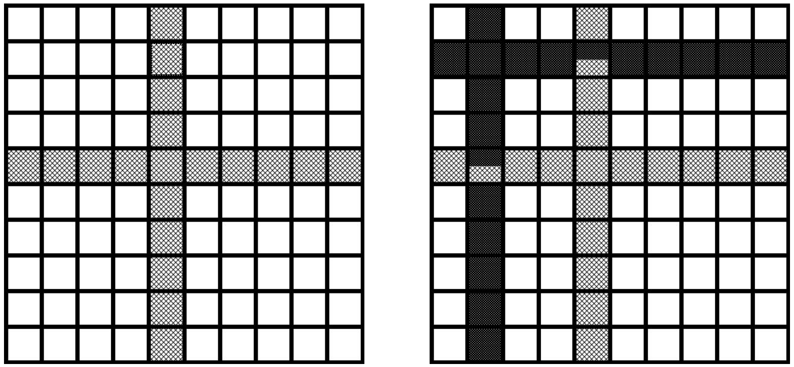


Majority quorum system
(all sets of n / 2 + 1 nodes)

	Singleton	Majority
How many servers need to be contacted? **(Work)**	1	> n/2
What's the load of the busiest server? **(Load)**	100%	≈ 50%
How many server failures can be tolerated? **(Resilience)**	0	< n/2

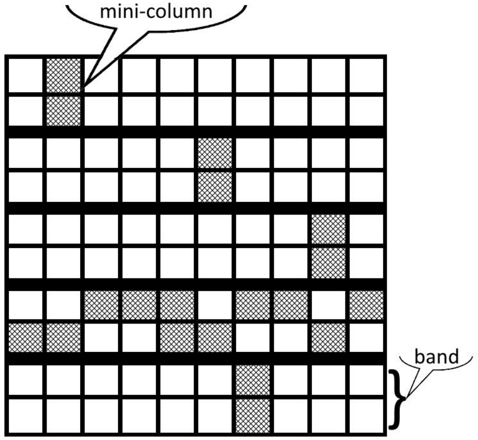
Basic Grid Quorum System

- Nodes arranged in a square matrix
- Each quorum i contains the union of row i and column i



B-Grid Quorum System

- Nodes arranged in rectangular grid with h·r rows
- Group of r rows is a band
- Group of r elements in the same column and band is a mini-column
- **Quorums** consists of one mini-column in every band and one element from each mini-column of one band



Quiz

1. Does a quorum system exist which can tolerate that all nodes of a specific quorum fail?

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

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

Quiz Solution

1. Does a quorum system exist which can tolerate that all nodes of a specific quorum fail?

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

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

Quiz Solution

1. Does a quorum system exist which can tolerate that all nodes of a specific quorum fail?

   A: no, as any two quorums intersect!

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

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

Quiz Solution

1. Does a quorum system exist which can tolerate that all nodes of a specific quorum fail?

   A: no, as any two quorums intersect!

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

   A: one, as two nodes failing fails all quorums!

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

Quiz Solution

1. Does a quorum system exist which can tolerate that all nodes of a specific quorum fail?

   A: no, as any two quorums intersect!

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

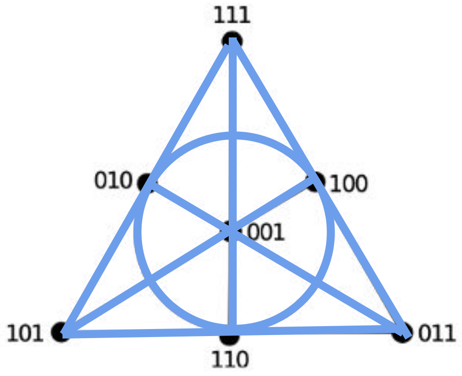
   A: one, as two nodes failing fails all quorums!

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

   A: pick a node and take all quorums containing it. Maximality: between any quorum and its complement at most one can be in the system.

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.



- Quorums: 7
- Work: 3
- Load: 3/7

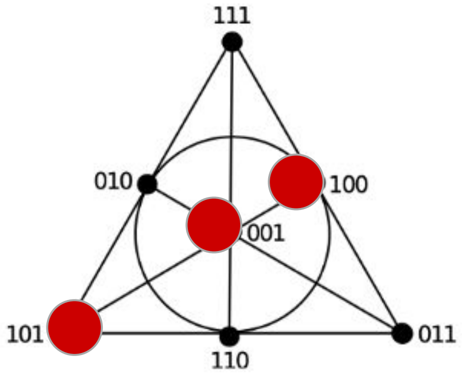
a) Of how many different quorums does this system consist and what are its work and its load?

A Quorum System

Systems@ETH zürich

Distributed Computing

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.



Resilience: 2

Every node is in 3 quorums
=> any two nodes can be contained in at most 2*3 quorums

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

ETH zürich · Systems@ETH zürich · Distributed Computing

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.

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.

Idea: No matter which quorum gets accessed, **exactly s nodes have to work**.
=> the sum of all loads should be to s

To minimize the maximum element of a sum, set all elements to the average (balanced access strategy).

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.

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

Uniform Quorum Systems

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:

$$\sum_{v_i \in V} L_Z(v_i) = \sum_{v_i \in V} \sum_{Q \in \mathcal{S}; v_i \in Q} P_Z(Q) = \sum_{Q \in \mathcal{S}} \sum_{v_i \in Q} P_Z(Q)$$
$$= \sum_{Q \in \mathcal{S}} P_Z(Q) \sum_{v_i \in Q} 1 \stackrel{*}{=} \sum_{Q \in \mathcal{S}} P_Z(Q) \cdot s = s \cdot \sum_{Q \in \mathcal{S}} P_Z(Q) = s$$

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

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