ETH

Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Systems@ETH Zürich

Distributed Systems in practice
Recitation Class 2 – 3PC/Quorum Systems

René Müller, Systems Group, ETH Zurich

muellren@inf.ethz.ch, IFW B49.1

HS 2008

Important Note: Download of the Book

- Apparently, Microsoft Research updated their website so the link to Phil Bernstein's Book "Concurrency Control and Recovery in Distributed Databases" is no longer valid.

- However, the FTP link (still) works.

- Alternatively, you can find the book on the VS_Wiki used earlier in the lecture.

Problems with 2PC

- In 2PC any process can block during its uncertainty period.
- However, if all processes are uncertain they all remain blocked.
  - Coordinator failed after deciding (coordinator is no longer uncertain)
- Issue is addressed in 3PC

Non-blocking Rule

- **NB**: If any operational process is uncertain then no process can have decided to commit.

- Solution to previous problem:
→ If all operational processes and find out that they are uncertain, they can safely abort, knowing that none of the failed processes could have decided commit.

Non-Blocking Rule in 3PC

- Idea: Use additional round of messages (**PRE-COMMIT**, **ACK**) to get everybody out of the uncertainty window.
- 3PC Coordinator sends **PRE-COMMIT** before **COMMIT**
- Semantics of **PRE-COMMIT**: Decision is going to be commit if there are no failures.
- A node receiving a **PRE-COMMIT** replies with an **ACK**.
- What's the purpose of the message? Coordinator has to **expect an ACK from each participant**.
- To signal an event! Signals that participant is participating in second phase

Three-Phase Commitment Protocol (3PC)

Roles

- Coordinator (C): initiates 3PC
- Participants (P)

Messages

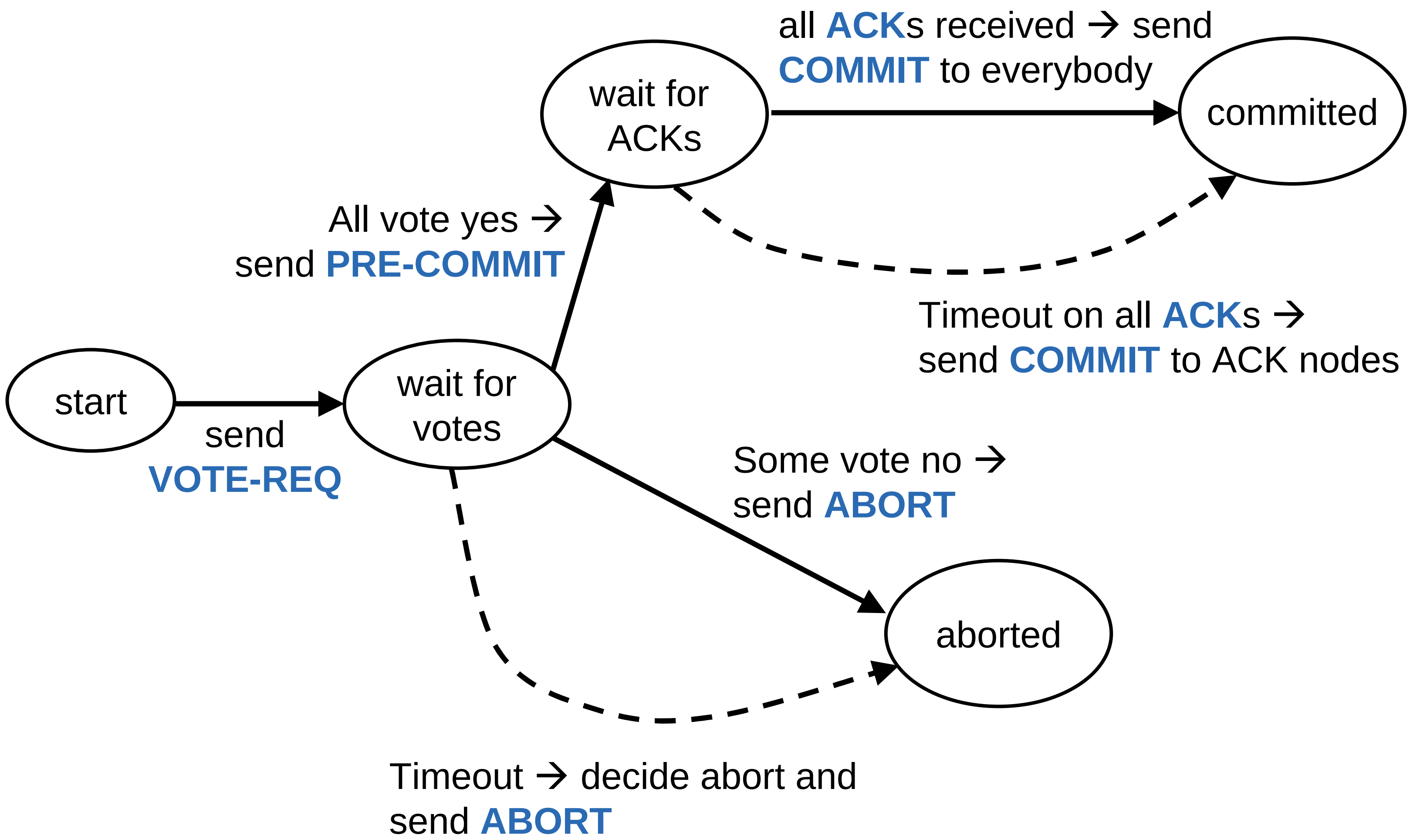
- **VOTE-REQ**: (C)→(P)
- **YES**, **NO**: (P)→(C)
- **PRE-COMMIT** (C)→(P)
- **ACK** (C)→(P)
- **COMMIT**, **ABORT** (C)→(P)

Timeouts on

- (P) **VOTE-REQ** → abort
- (C) **YES**, **NO** → abort
- (P) **PRE-COMMIT** → term. prot.
  (C) **ACK** → ignore failed Ps
- (P) **COMMIT** → term. protocol

1. Coordinator sends **VOTE-REQ** to all participants.
2. When receiving **VOTE-REQ** participant votes and sends **YES**/**NO** vote to coordinator.
3. Coordinator collects votes and decides commit/abort.
   - All vote yes → **PRE-COMMIT**
   - Otherwise → **ABORT**
4. Participants receive
   1. **PRE-COMMIT** reply **ACK**
   2. **ABORT** → abort
5. Coordinator receives **ACK**s then sends **COMMIT** to those it received an **ACK** from.

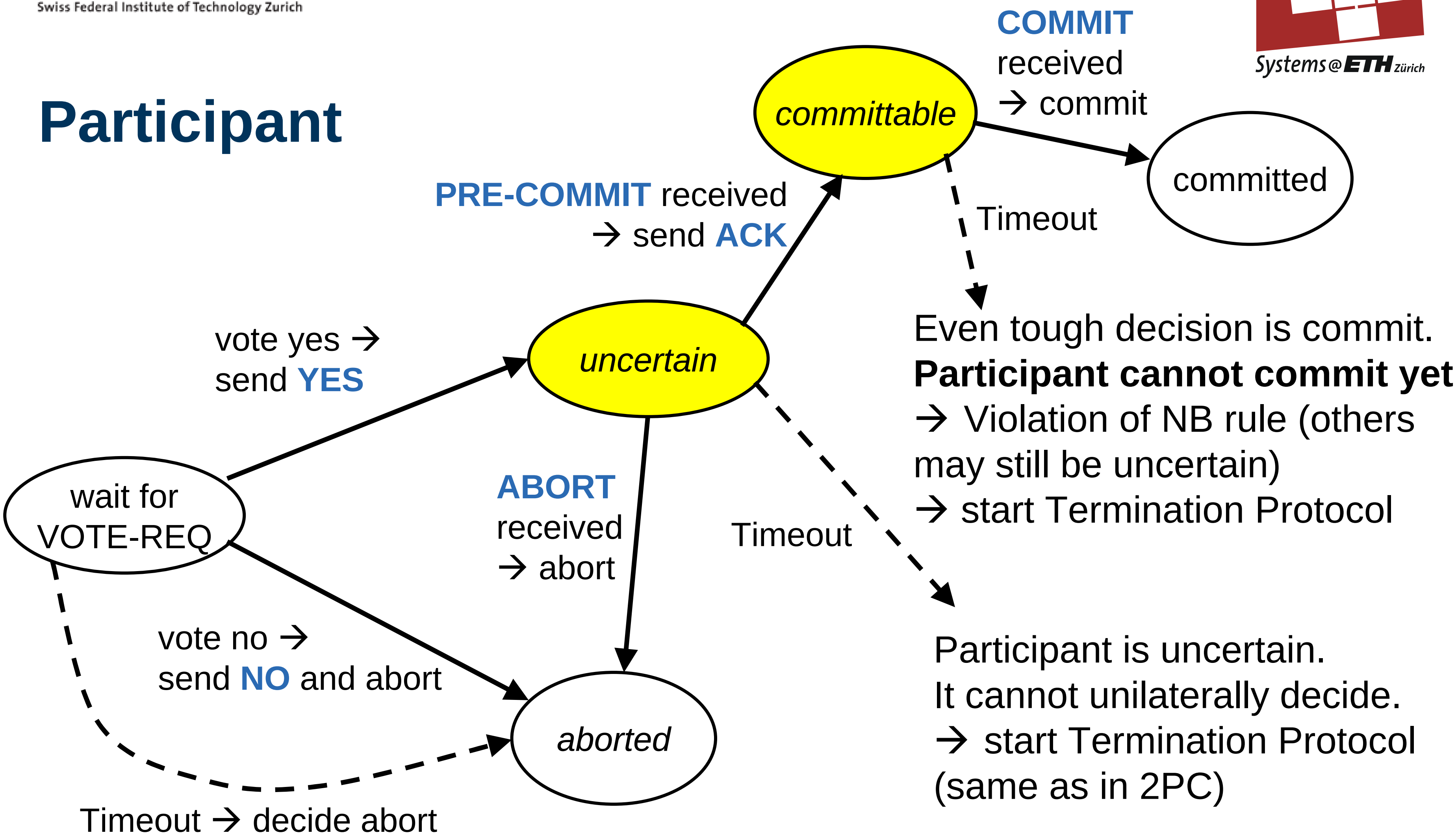
Coordinator



ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Systems@ETH zürich

Participant



Termination Protocol

1. Elect new coordinator
2. Coordinator sends **STATE-REQ** to all processes in the election.
3. All operating processes report their state
4. Coordinator applies **Termination Rules** based on state reports:

TR1: If some process is *aborted* → send **ABORT**

TR2: If some process is *committed* → send **COMMIT**

TR3: If some process is *uncertain* → decide abort and send **ABORT**.

TR4: If some processes is *committable* but none is *committed* → resume 3PC as new coordinator by (re-)sending **PRE-COMMIT**.

Coexistence of States

	Aborted	*Uncertain*	*Committable*	*Committed*
Aborted	✓ TR1	✓ TR3	✗	✗
Uncertain	✓	✓ TR3	✓ TR3	✗
Committable	✗	✓	✓ TR4	✓ TR2
Committed	✗	✗	✓	✓ TR2

→ For each feasible combination there is exactly one termination rule

Failures in 3PC

- Fact: Logging **PRE-COMMIT** and **ACK**s does not help in recovery.
  - → Logging identical to 2PC.
- Recovery from total site failures
  - wait for last process that failed (unless independent recovery possible) → termination protocol must include last failing process.
- Communication failures
  - Partitioning can occur
  - Partition may decide differently → inconsistency
  - Protocol does **NOT** tolerate communication failures.
  - *Solution*: Use Quorums, i.e. decide only when majority of processes are participating. → introduces blocking again, of no quorum can be obtained.

Assignment 7.14

	Aborted	*Uncertain*	*Committable*	*Committed*
Aborted	✓ (1)	✓ (2)	✗ (3)	✗ (4)
Uncertain		✓ (5)	✓ (6)	✗ (7)
Committable			✓ (8)	✓ (9)
Committed				✓ (10)

Prove correctness of co-existence table.

(symmetry → only 10 cases)

Coexistence Table: simple cases

(1) ***Aborted—Aborted***: no failures, a **NO** vote → abort.

(2) ***Aborted—Uncertain***: p_1 votes **NO** and unilaterally aborts, p_2 votes yes and is uncertain.

(5) Uncertain—*Uncertain*: p_1 and p_2 vote YES, however, do not yet know the decision made by the coordinator.

(6) Uncertain—*Committable*: after situation (5) the coordinator sends **PRE-COMMIT**. p_1 received it before p_2 → p_1 committable while p_2 still uncertain.

(7) ***Uncertain—Committed***: prevented by NB rule. When committed there are no operational uncertain processes.

(8) ***Committable—Committable***: step (6) after p_2 got **PRE-COMMIT**

(9) ***Committable—Committed***: p_2 has received **COMMIT** p_1 not yet.

(10) ***Committed—Committed***: step (6) after p_1 also received **COMMIT**.

Coexistence Table: remaining cases

(3) *Aborted—Committable*

(no communication failures)

Abort possible if

- In termination protocol when Committable $\Rightarrow$ everybody voted yes
- Hence, processes are either uncertain or committable.
- Abort then only in termination protocol.
- Consider first round that would decide abort
  - Abort if some are uncertain processes are operational → **impossible** (no communication failures)

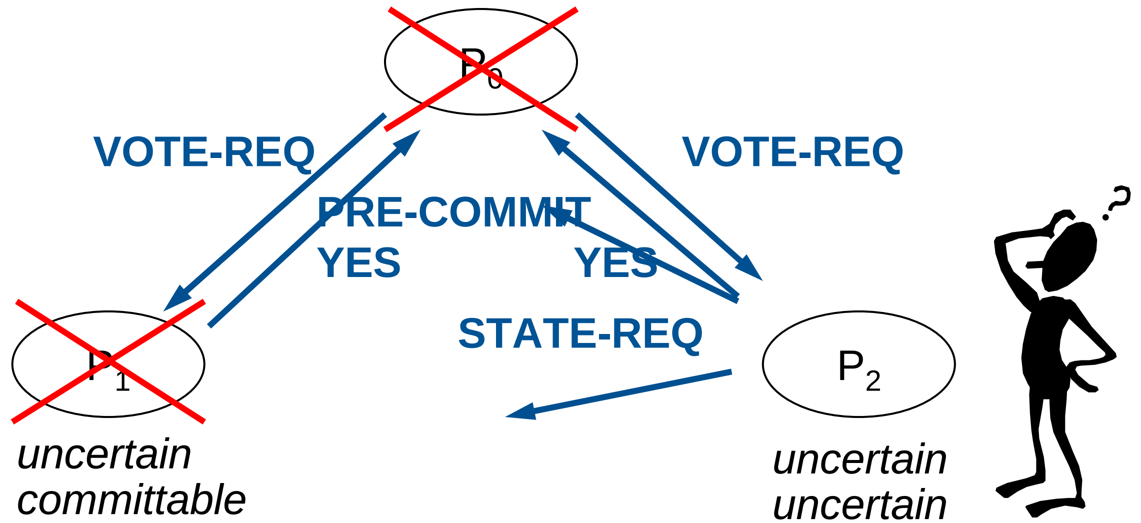
(4) *Aborted—Committed*

Commit is only reached if committable before.

However, (3) says impossible

Assignment 7.17

- Describe scenario with site-failures only where a *committable* process still would lead to an abort.



termination protocol
"I am the only one alive and uncertain so I abort"

Assignment 7.17

1. P_0 sends **VOTE-REQ** to P_1 and P_2
2. P_1 and P_2 both reply with **YES**
3. P_0 sends **PRE-COMMIT** to P_1 but fails before sending it to P_2. Thus, P_1 is committable whereas P_2 is still uncertain.
4. P_1 fails.
5. P_2 times out for the **PRE-COMMIT** and starts termination protocol.
6. P_2 sends out **STATE-REQ**.
7. P_2 times out for replies and since it is the only one alive, determines abort since it is uncertain.

Assignment 3 (a)

- Read One-Write All (ROWA) Systems
  - Advantage cheap reads: one local read
  - Disadvantage expensive writes: N writes
- ROWA suitable for read-dominated loads

- Apparent trade-off: read costs ⇔ write costs

- Synchronous Update Everywhere ROWA: cheap reads expensive writes
- Asynchronous Update Primary Copy: cheap writes expensive reads (local read may be out-of-date)

- Is there something in-between, i.e., not write-all and read “a few”?

Quorum Systems

- Improve performance with availability in replication.
- Balance costs between read and write operations.
- Reduce number of copies involved in updates

- Beispiel aus der Politik: “Für Verhandlungs- und Beschlussfähigkeit der vereinigten Bundesversammlung ist die Anwesenheit von mehr als der Hälfte (>50%) der Räte erforderlich. “ → Dann “absolutes Mehr”.

Types
- Voting Quorums
  - **Majority Quorum** (Quorum Consensus, “Gewichtetes Votieren”)
  - Hierarchical Quorum Consensus
- Grid Quorums
- Tree Quorums

Quorums

Formal Definition:

- A **quorum system** $S = \{S_1, S_2, \ldots, S_N\}$ is a collection of **quorum sets** $S_i \subseteq U$ of a finite universe.
- $\forall\ i,j \in \{1, \ldots, N\} : S_i \cap S_j \neq \varnothing$.

- For replication we consider two quorum sets: **read quorum** RQ and **write quorum** WQ.

- Rules
  - Any read quorum must overlap with any write quorum
  - Any two write quorum must overlap

Majority Quorum

- Use vote to define quorum
- Each site has a non-negative voting weight.
- Majority = number of votes exceed half of the total votes

- For Assignment 3
  - For simplicity, we assume each site has vote weight 1.
  - N is the number of sites
  - Let |S| denote the voting weight of a quorum set S.

- Rules for read quorum (RQ) and write quorum (WQ)
  - $|RQ| + |WQ| > N \Rightarrow$ read and write quorums overlap
  - $2\,|WR| > N \Rightarrow$ two write quorums overlap

Quorum Sizes

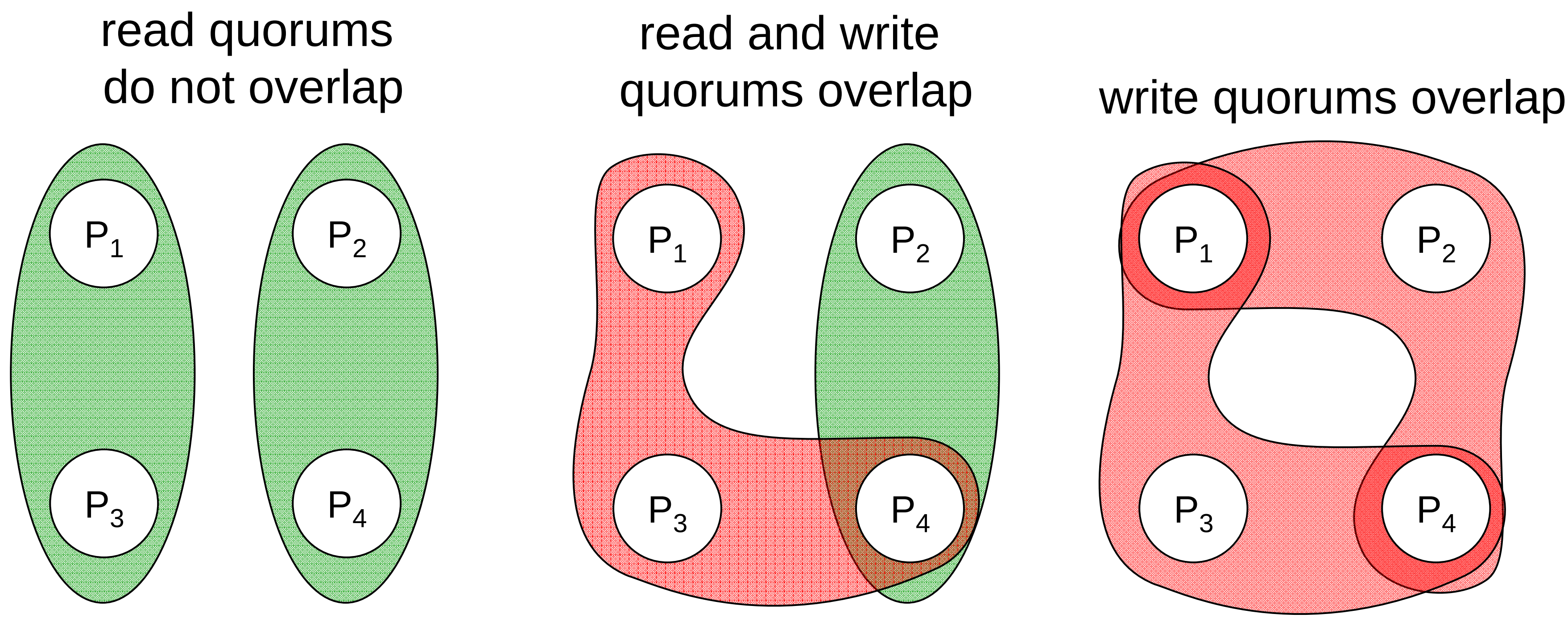
- Rules for read quorum (RQ) and write quorum (WQ)
  - |RQ| + |WQ| > N $\Rightarrow$ read and write quorums overlap
  - 2 |WR| > N $\Rightarrow$ two write quorums overlap
- The quorum sizes |RQ| and |WQ| determines the cost for read and write operations. → minimize!
- Minimum quorum sizes for the inequalities are:

$$\min\ |WQ| = \left\lfloor \frac{N}{2} \right\rfloor + 1 \qquad \min\ |RQ| = \left\lceil \frac{N}{2} \right\rceil$$

  - Write quorum requires majority
  - Read quorum requires at least half of the system sites

Example

- Consider 4 sites
  - min |WQ|=3 sites (majority)
  - min |RQ|=2 sites (half)



Comparison with ROWA

- For ROWA we can think of:
- |RQ| = 1 and |WQ|=N.

- Any read overlaps with any write
- Any two writes overlap
- Reads do not overlap

- For Quorums: $|WQ| \geq \left\lfloor \frac{N}{2} \right\rfloor + 1$ $\qquad |RQ| \geq \left\lceil \frac{N}{2} \right\rceil$

Assignment 3 (b)

- Load consists of R reads and W writes
  - Normalized: R+W=1

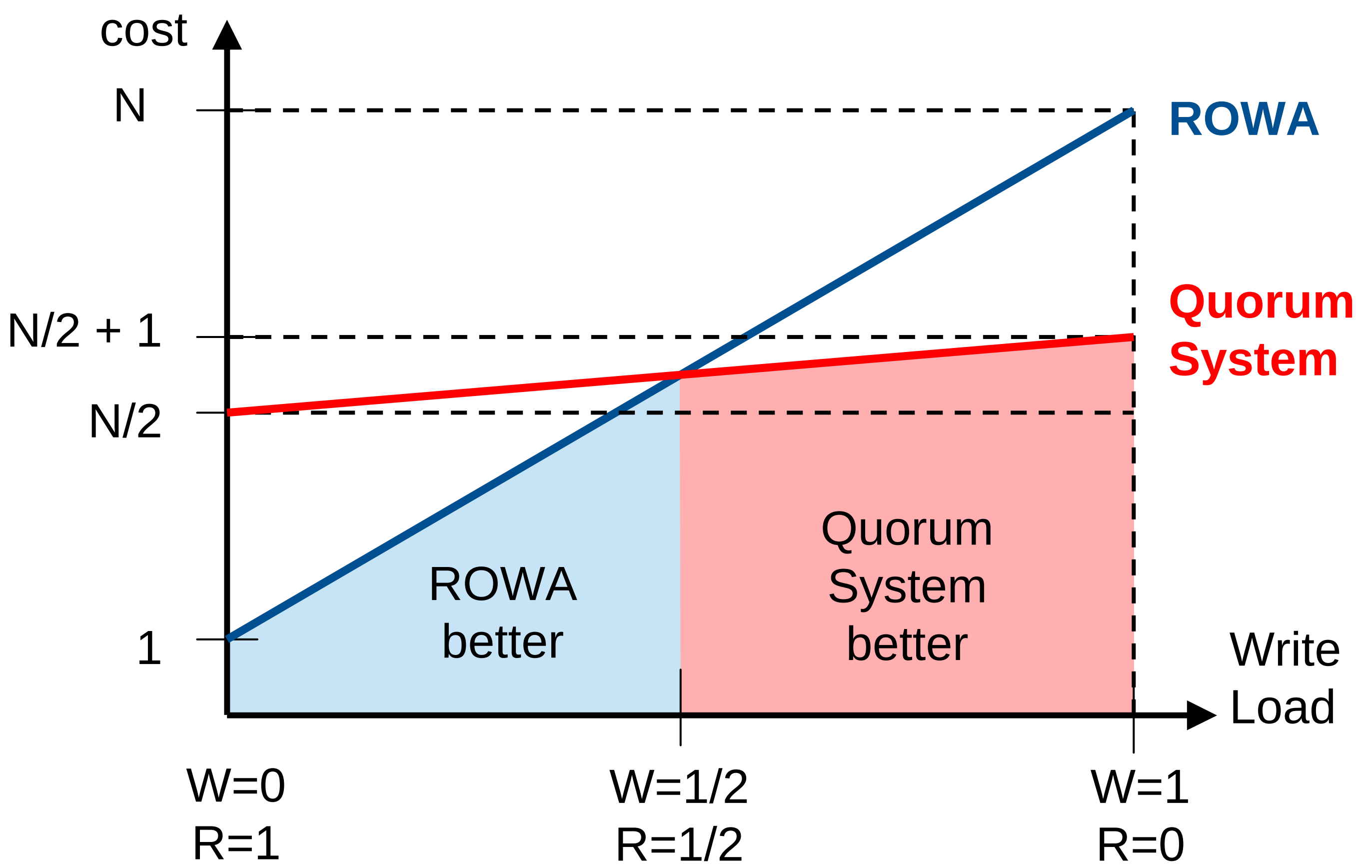
- Cost ROWA = R + N × W

- Cost Quorum = R × |RQ| + W × |WQ|

- For Minimum-sized quorums

$$\text{Cost} = R \times \left\lceil \frac{N}{2} \right\rceil + W \times \left(\left\lfloor \frac{N}{2} \right\rfloor + 1 \right)$$

ROWA – Quorum System



Assignment 3 (c)

- *Why has asynchronous replication lower cost than synchronous replication?*

- Cost for synchronous ROWA is
  Cost ROWA = R + N $\times$ W

- In terms of read/write operations asynchronous (primary copy) has cost 1
  → one direct write (master)
  → one local read (possibly outdated copy)
  → load independent

Updates

- However, this is not the full cost.
- Cost for propagating update sets (and reconciliation) also need to be considered.

- Assume, updates are load-independent with update frequency (rate r)
- Cost = 1 + r × (N-1)
- Thus, asynchronous, update primary copy is cheaper for

$$1 + r \times (N-1) \leq R + N \times W$$

$$r \leq \frac{R + N \times W - 1}{N - 1}$$

References

- R. Jiménez-Peris, M. Patiño-Martínez, G. Alonso, B. Kemme: **Are Quorums an Alternative for Data Replication?** ACM Transactions on Database Systems, 2003.

  http://doi.acm.org/10.1145/937598.937601