ΚΥΠΡΙΑΚΗ ΔΗΜΟΚΡΑΤΙΑ

ΕΥΡΩΠΑΪΚΗ ΕΝΩΣΗ

Η ΔΕΣΜΗ 2009-10 ΣΥΓΧΡΗΜΑΤΟΔΟΤΕΙΤΑΙ ΑΠΟ ΤΗΝ ΚΥΠΡΙΑΚΗ ΔΗΜΟΚΡΑΤΙΑ
ΚΑΙ ΤΟ ΕΥΡΩΠΑΪΚΟ ΤΑΜΕΙΟ ΠΕΡΙΦΕΡΕΙΑΚΗΣ ΑΝΑΠΤΥΞΗΣ ΤΗΣ ΕΕ

Seeking Fastness in Multi-Writer Multiple-Reader Atomic Register Implementations

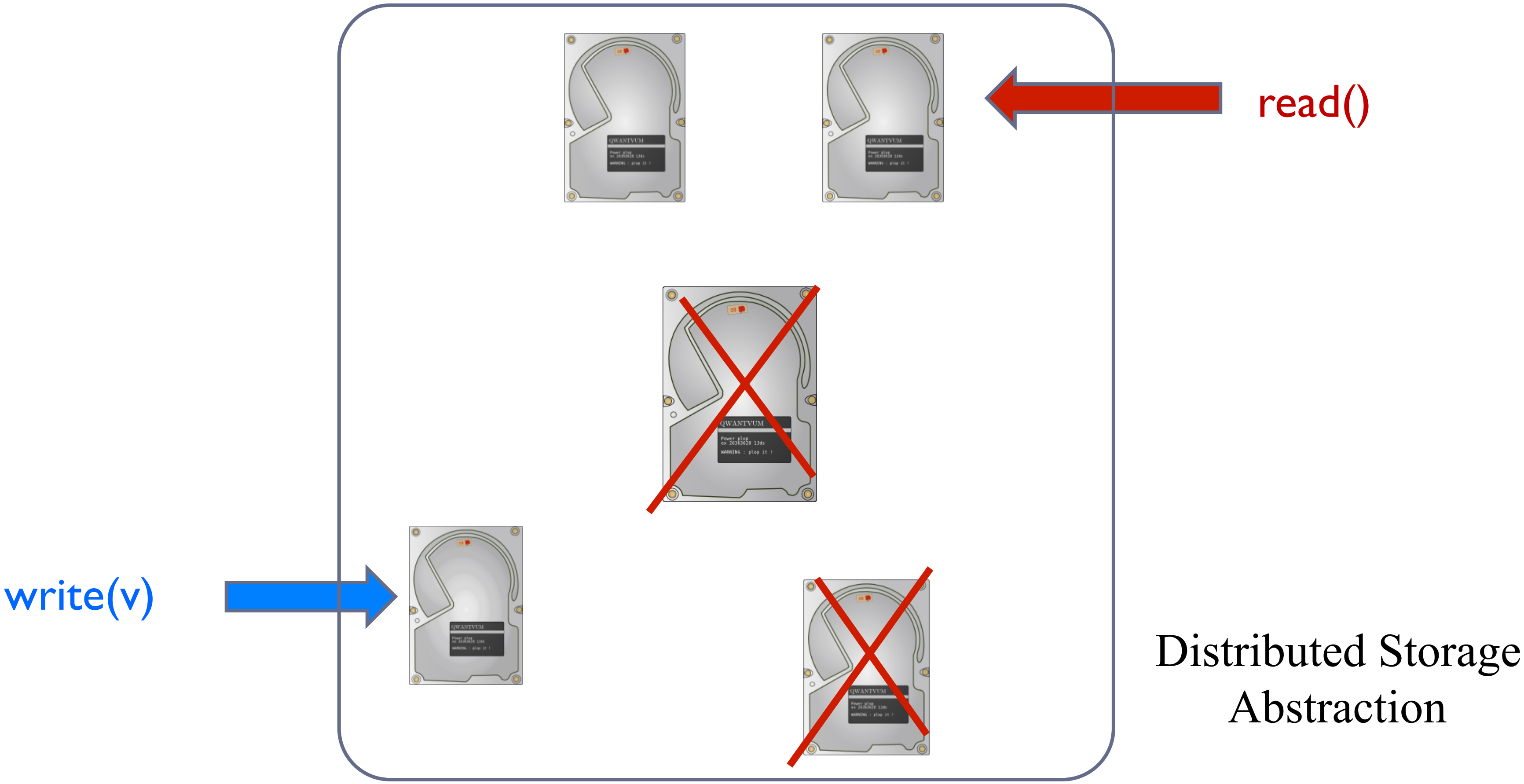
Π Ε Ν Ε Κ /0609/31

Nicolas Nicolaou

University of Cyprus &
University of Connecticut

Funded by the Cyprus Research Promotion Foundation and co-funded by the Republic of Cyprus and the European Regional Development Fund

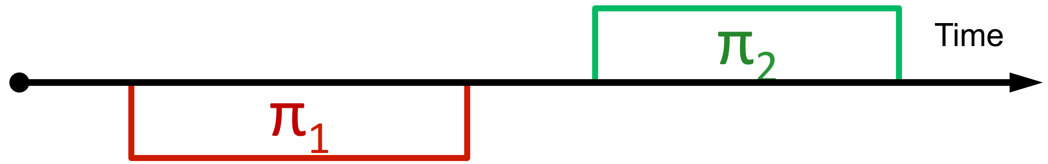
What is a Distributed Storage System?



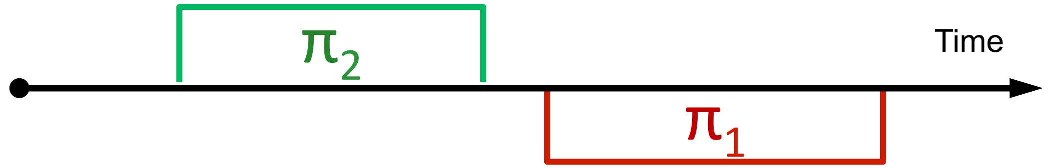
- Data Replication – Servers/Disks
  - Survivability and Availability
- Read/Write operations
- Consistency Semantics

Definition: Operation Relations

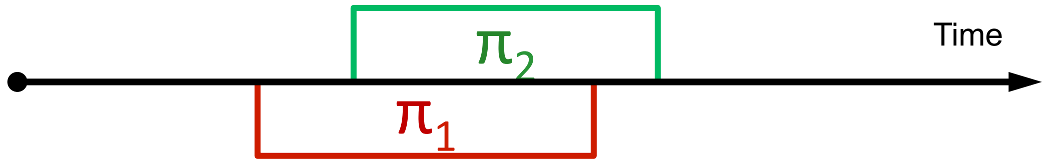
- Precedence Relations for two operations π_1 ,π_2:
  - π_1 precedes π_2 if the response of π_1 happens before the invocation of π_2



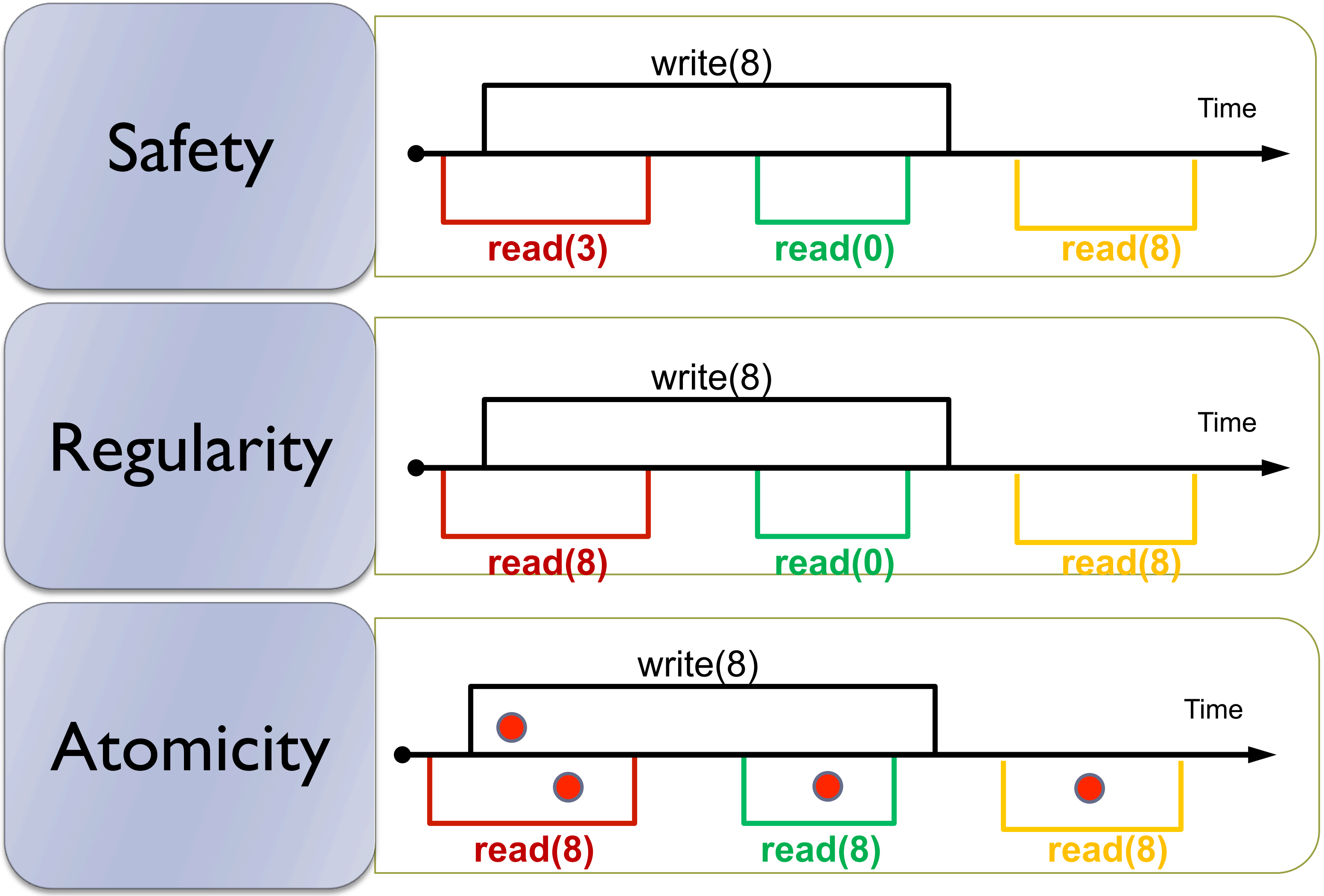
  - π_1 succeeds π_2 if the invocation of π_1 happens after the response of π_2



  - π_1 is concurrent with π_2 if π_1 neither precedes nor succeeds π_2



Consistency Semantics [Lamport86]



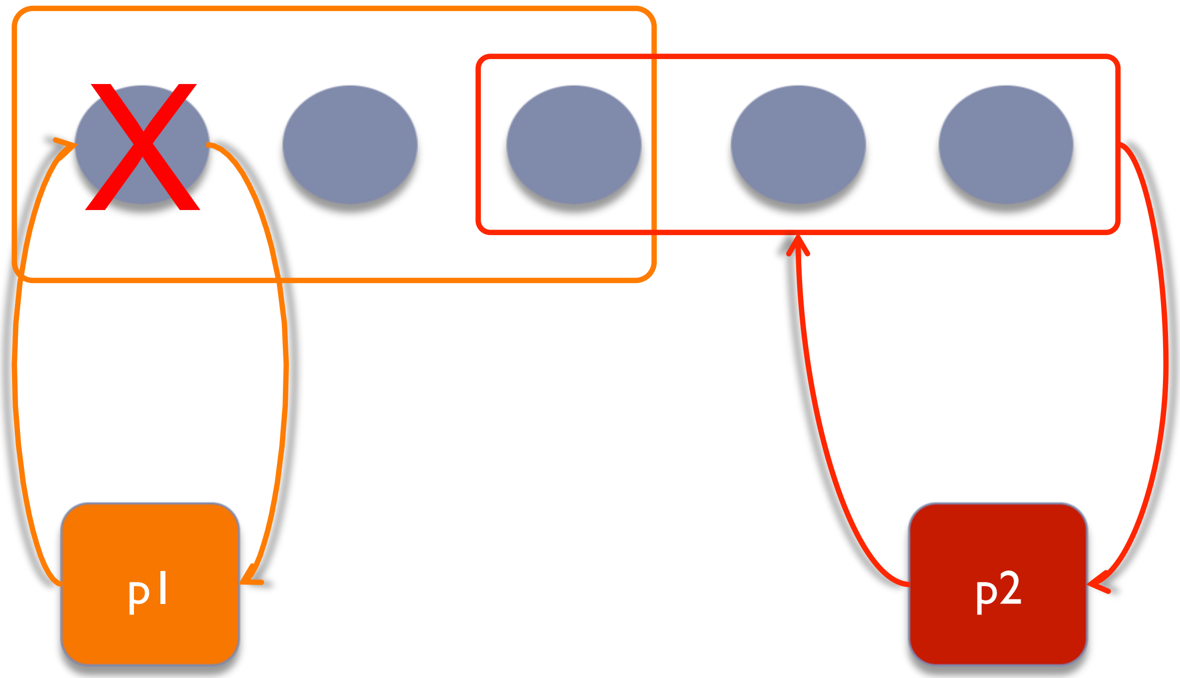
How to order read/write operations?

- Based on the value each operation writes/returns
  - Non-unique Values

- Using the “time” at which each operation is invoked
  - Clock Synchronization

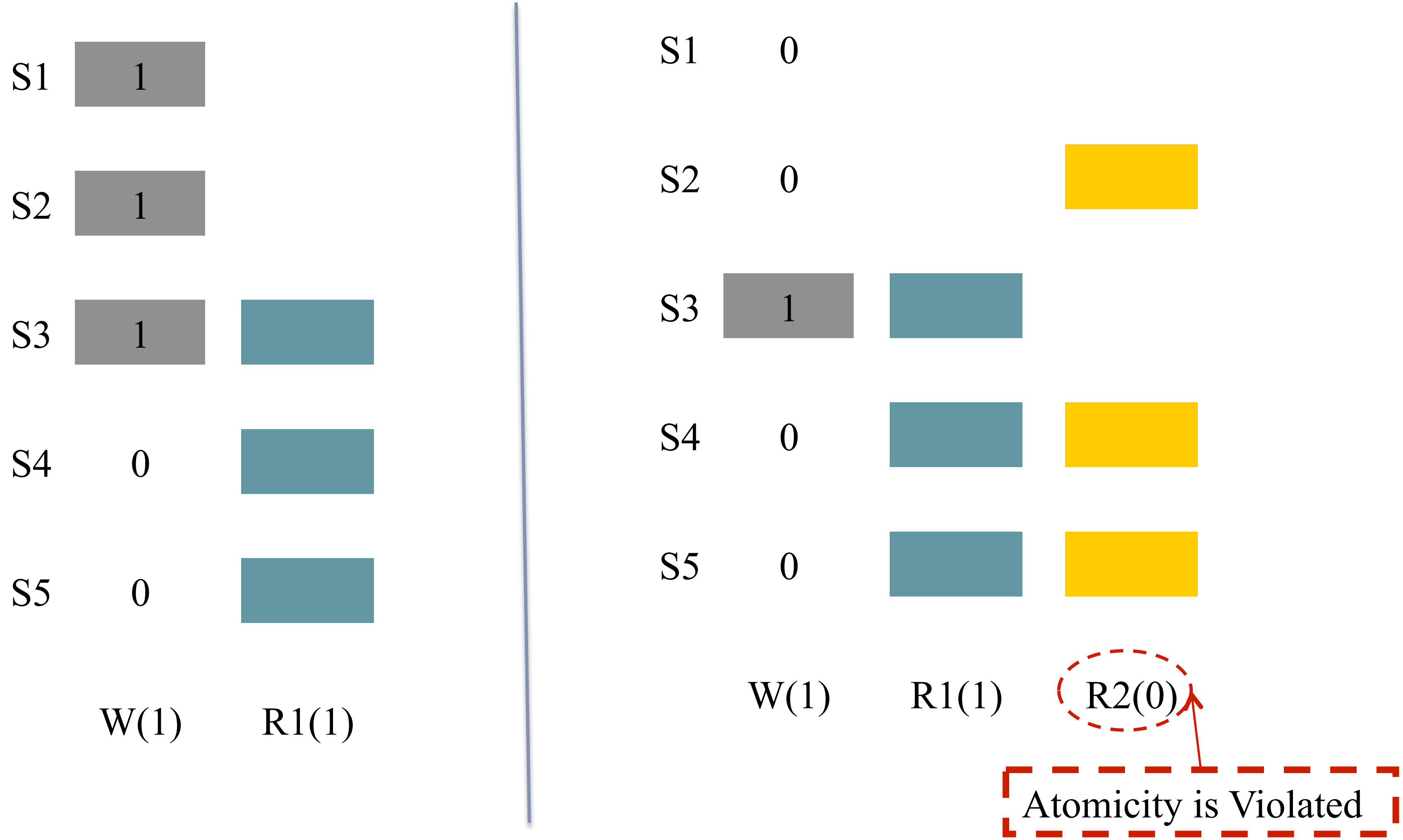
- Associate a sequence number with each value written
  - SWMR: timestamps
  - MWMR: tags=<timestamp, wid>

Challenges – Communication Rounds

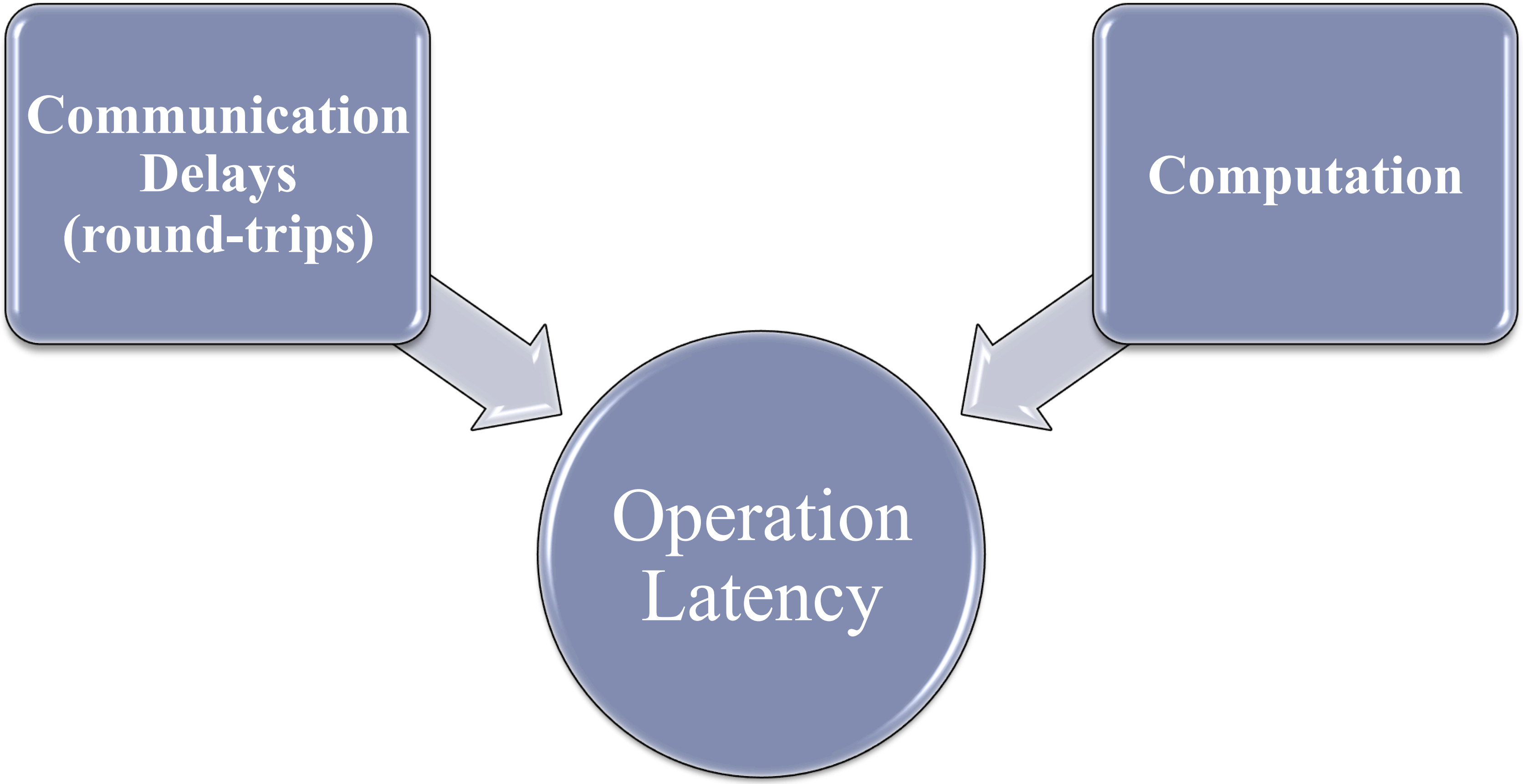


Multiple Round-Trips

- Consider the following example [Attiya et al. 96]:



Complexity Measure



What was known…

Traditional SWMR

- [Attiya et al. 95]
- Single round writes
- Two round reads
  - Phase 1: Obtain latest value
  - Phase 2: Propagate latest value
    - Folklore belief: "Reads must Write"

Traditional MWMR

- Two round writes
  - Phase 1: Discover latest value
  - Phase 2: Order new value after the latest and propagate
    - Belief: "Writes must Read"
- Two round reads

The Era of Fast Implementations…

SWMR Fast

- Single round (***fast***) writes and reads
  - Bounded readers: $R<(S/f)-2$ where S servers & f failures
  - Impossible in MWMR model

SWMR Semifast

- Fast writes
- Only a single complete 2-round (***slow***) read per write
  - Unbounded readers
  - Impossible in the MWMR model

SWMR Weak-Semifast

- General Quorum System
- Fast writes and Multiple slow reads per write
  - Allows concurrent fast reads with writes
  - Unknown if applicable in MWMR model

Model

- Asynchronous, Message-Passing model
  - Process sets: writers W, readers R, servers S (replica hosts)
  - Reliable Communication Channels
  - Well Formedness

- Environments:
  - SWMR: $|W|=1$, $|R|\geq 1$
  - MWMR: $|W|\geq 1$, $|R|\geq 1$

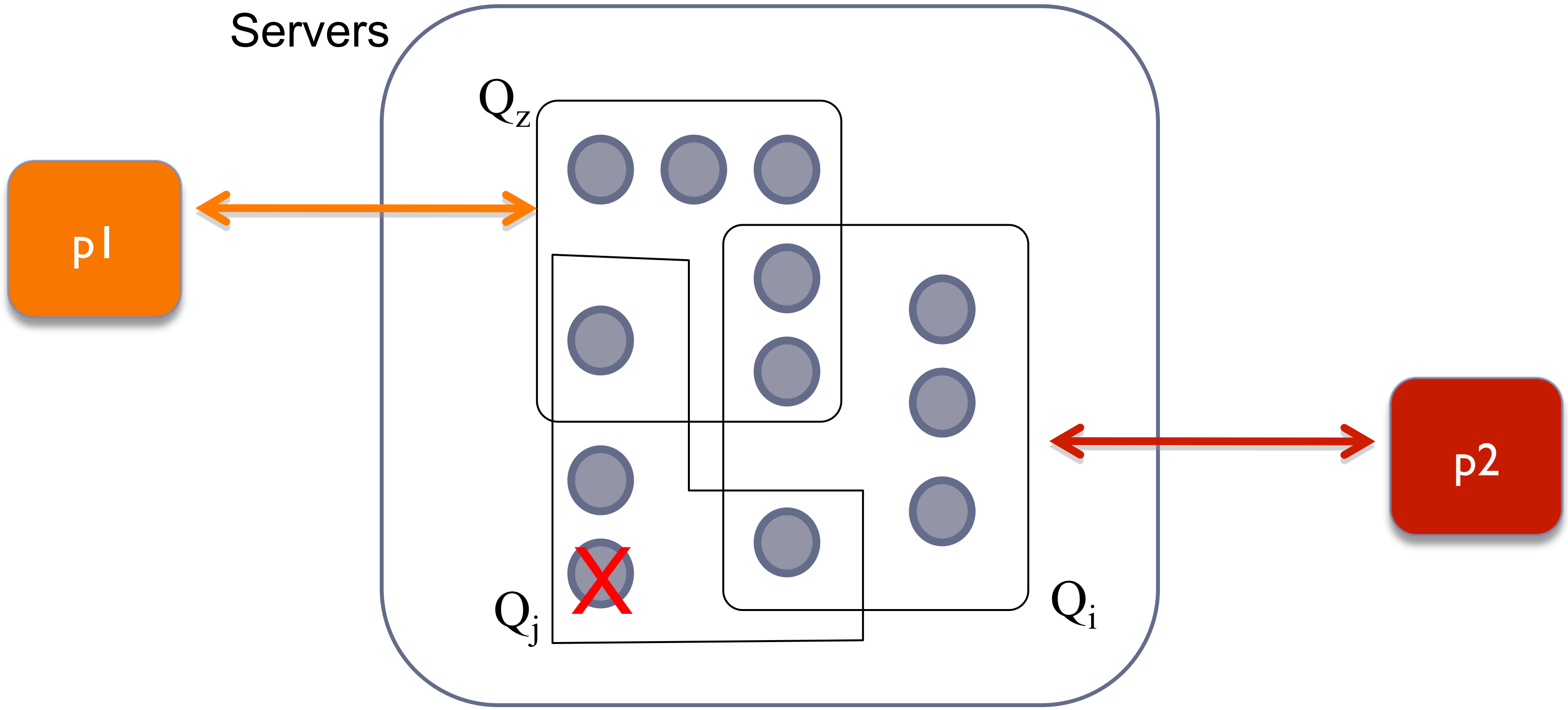
- Failures:
  - Crash Failures

- Correctness: Atomicity (safety), Termination (liveness)

Communication Round

- A process p performs a communication round during an operation π if:
  - p sends a message m to a set of servers for π
  - Any server that receives m replies to p
  - Once p receives responses from a single quorum completes π or proceeds to a next communication round

Definition: Quorum systems



- Q_i, Q_j, Q_z are **quorums**
- **Quorum System** is the set $\{Q_i, Q_j, Q_z\}$
  - Property: every pair of quorums intersects
  - N-wise quorums systems: every N quorums intersect for N>1
- Every R/W operation communicates with a single quorum
- Faulty Quorum: Contains a faulty process

Algorithm: Simple

Write Protocol: two rounds

- P1: Query a single quorum for the latest tag
- P2: Increment the timestamp in the max tag, and send <newtag, v> to a quorum

Read Protocol: two rounds

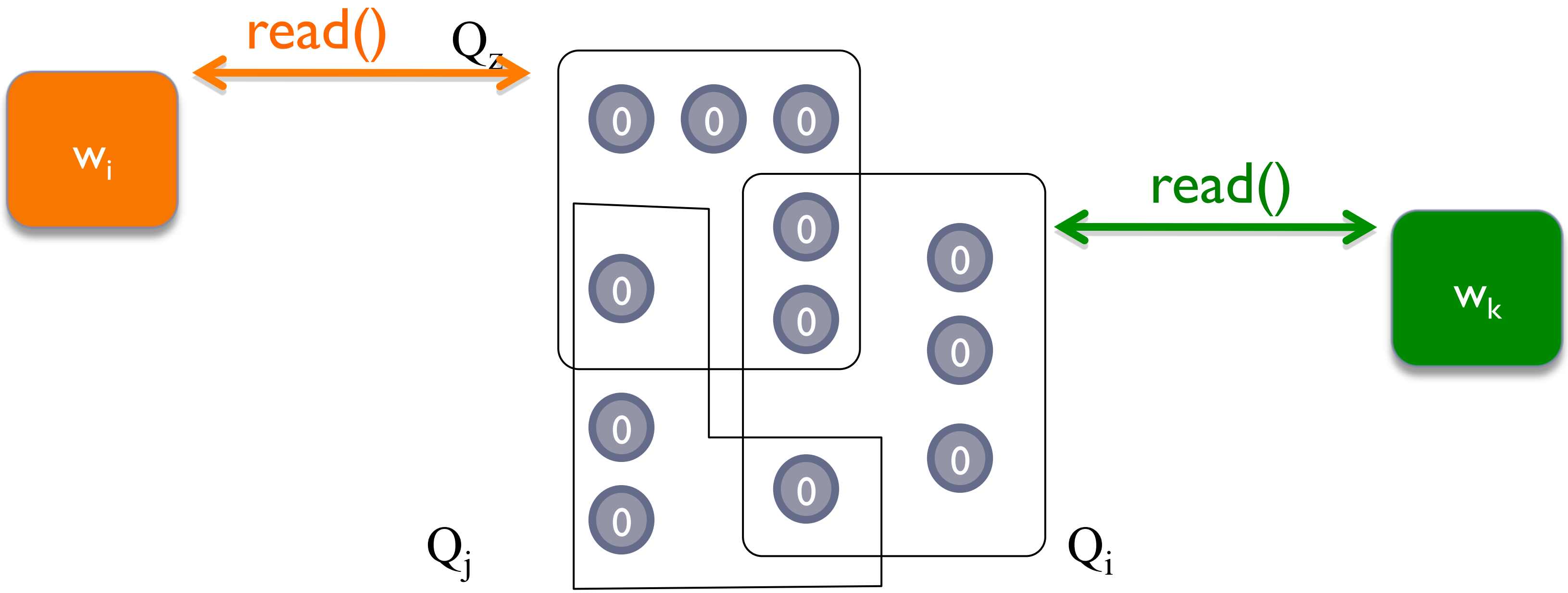
- P1: Query a single quorum for the latest tag
- P2: Propagate <maxtag,v> to a single quorum

Server Protocol: passive role

- Receive requests, update local timestamp (if msg.tag>server.tag) and reply with <server.tag,v>

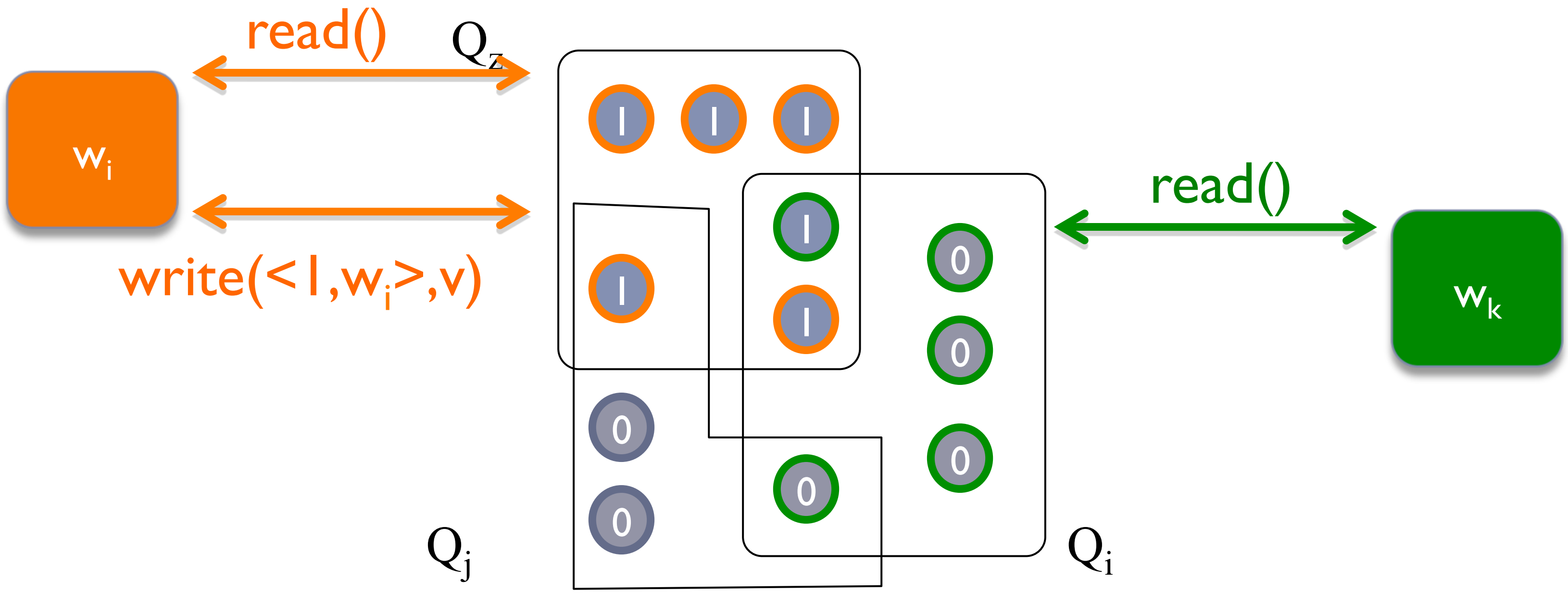
Example: Simple (write operations)

- Assume $w_i > w_k$



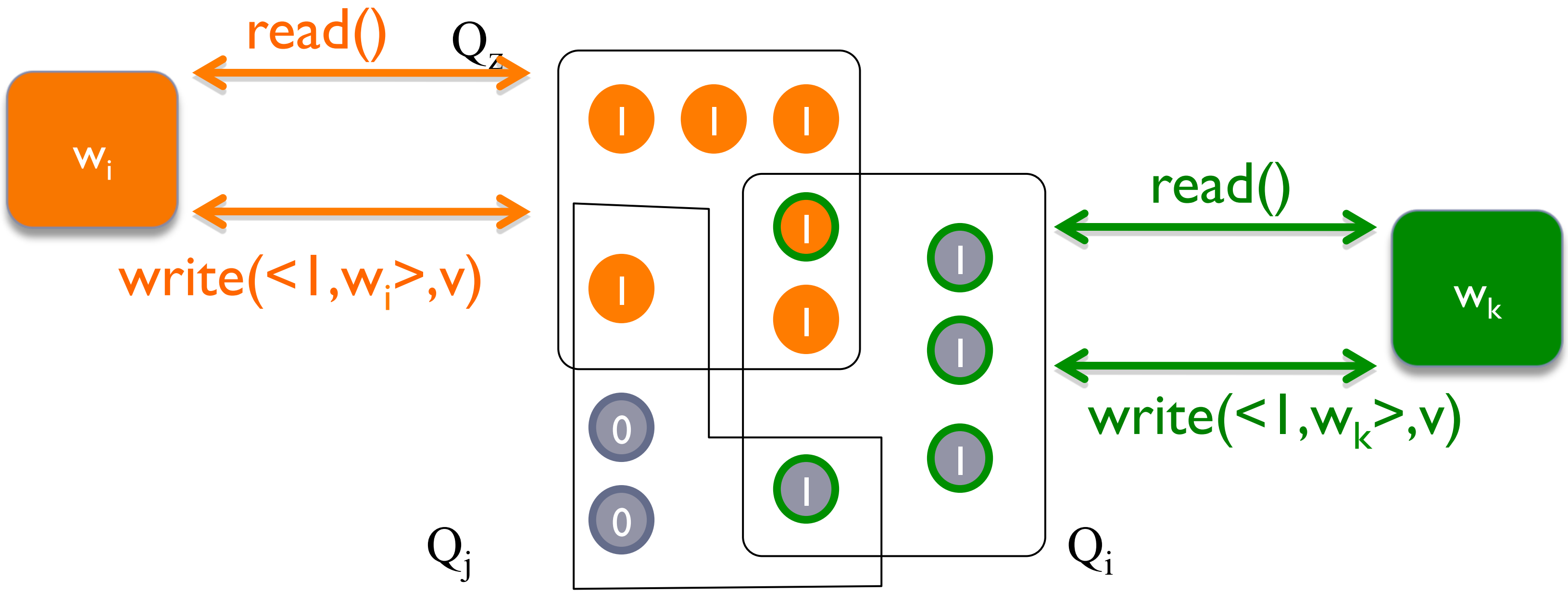
Example: Simple (write operations)

- Assume $w_i > w_k$



Example: Simple (write operations)

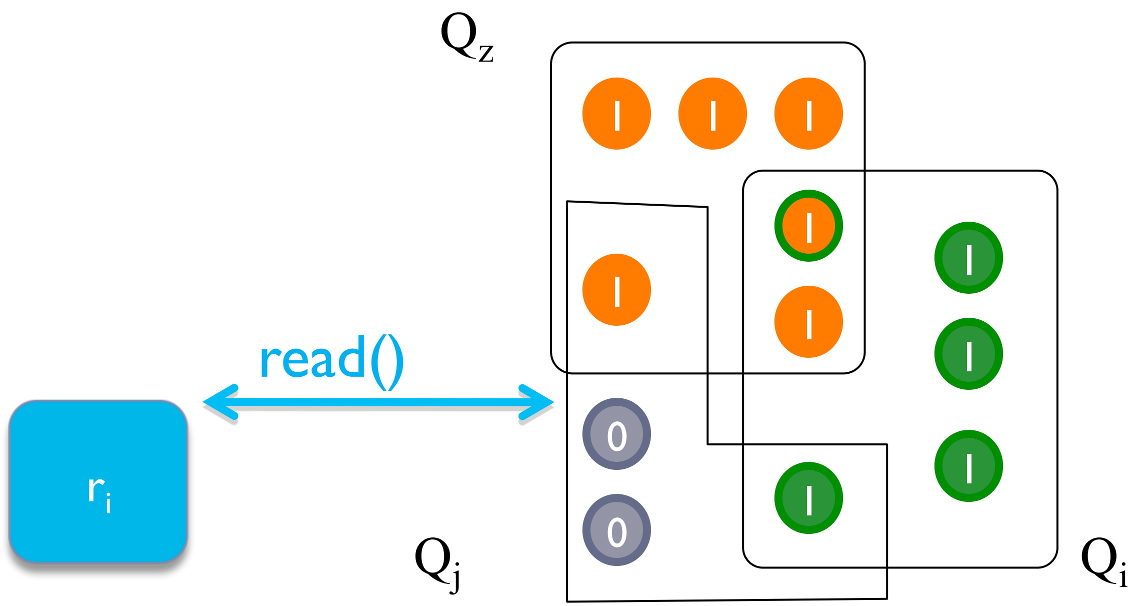
- Assume $w_i > w_k$



Belief: **Writes must Read in MW environments**

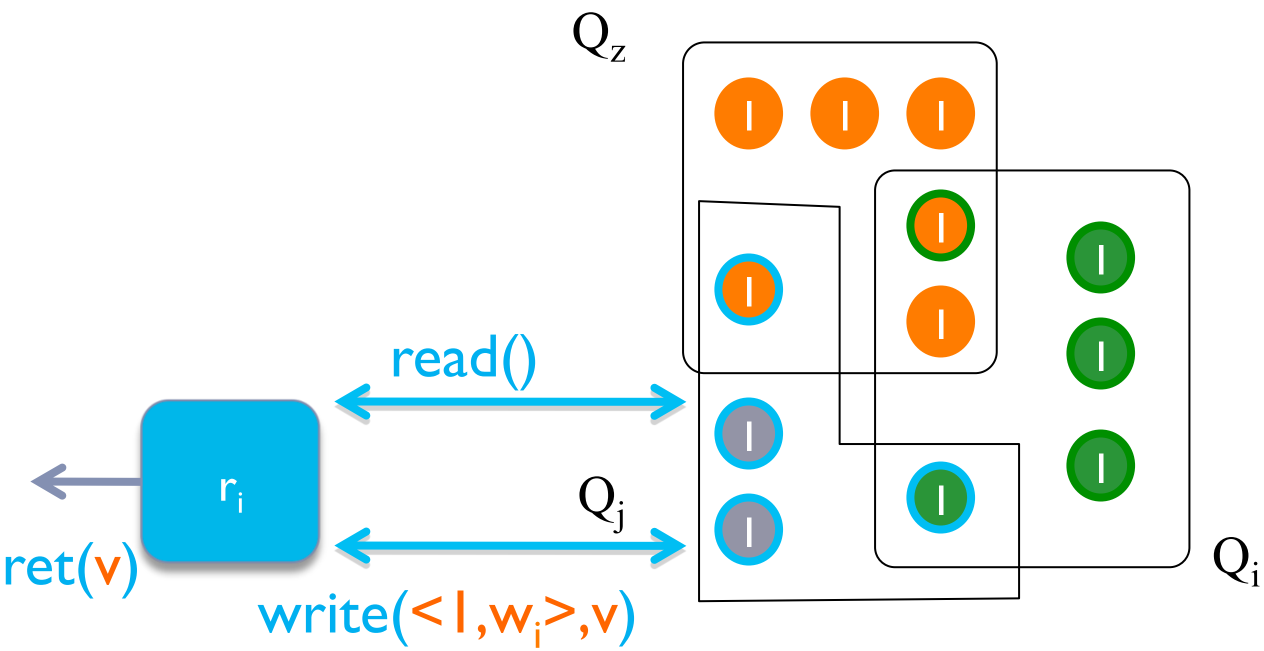
Example: Simple (read operation)

- Assume $w_i > w_k$



Example: Simple (read operation)

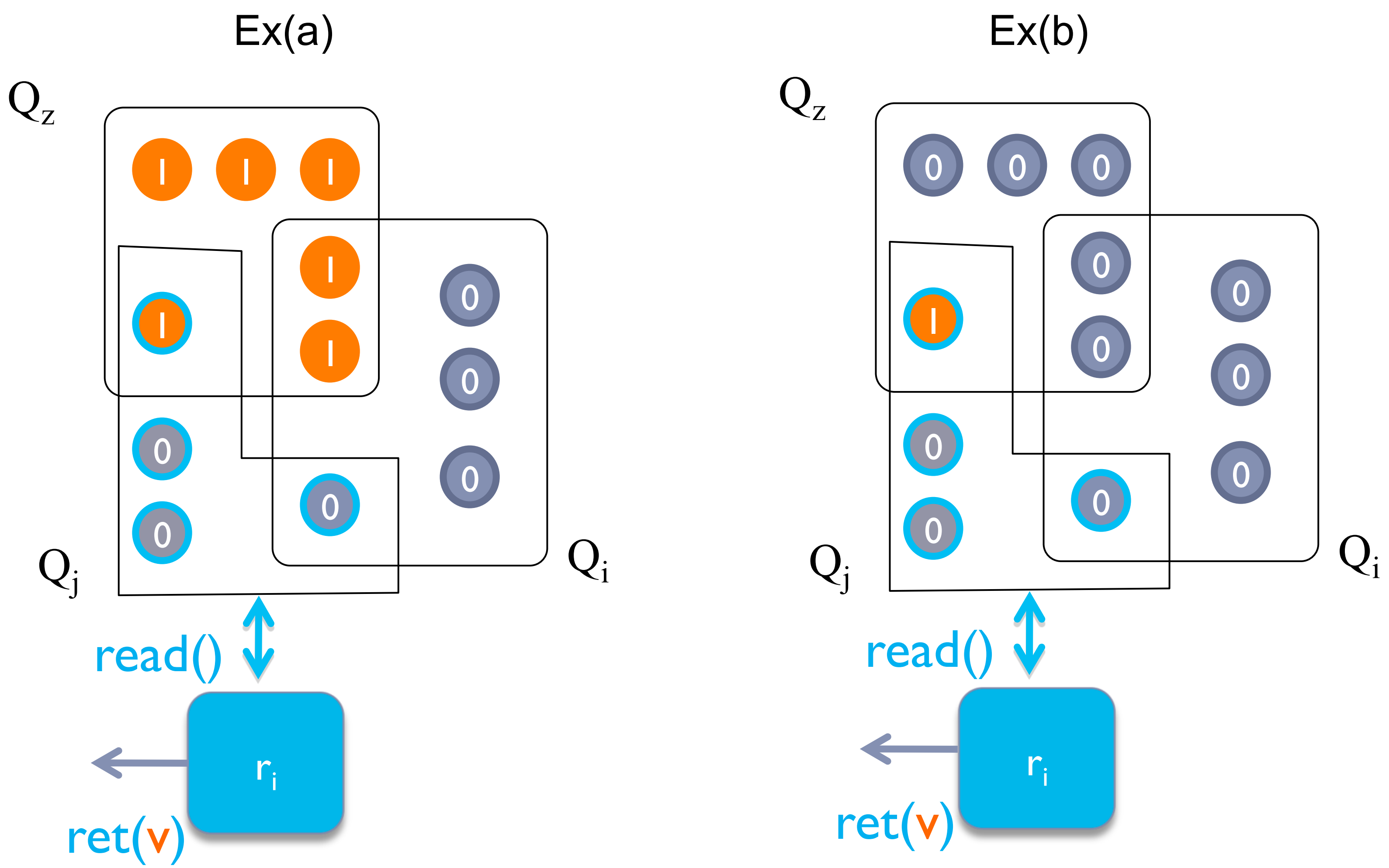
- Assume $w_i > w_k$



Operation Ordering: w_k -> w_i -> r_i

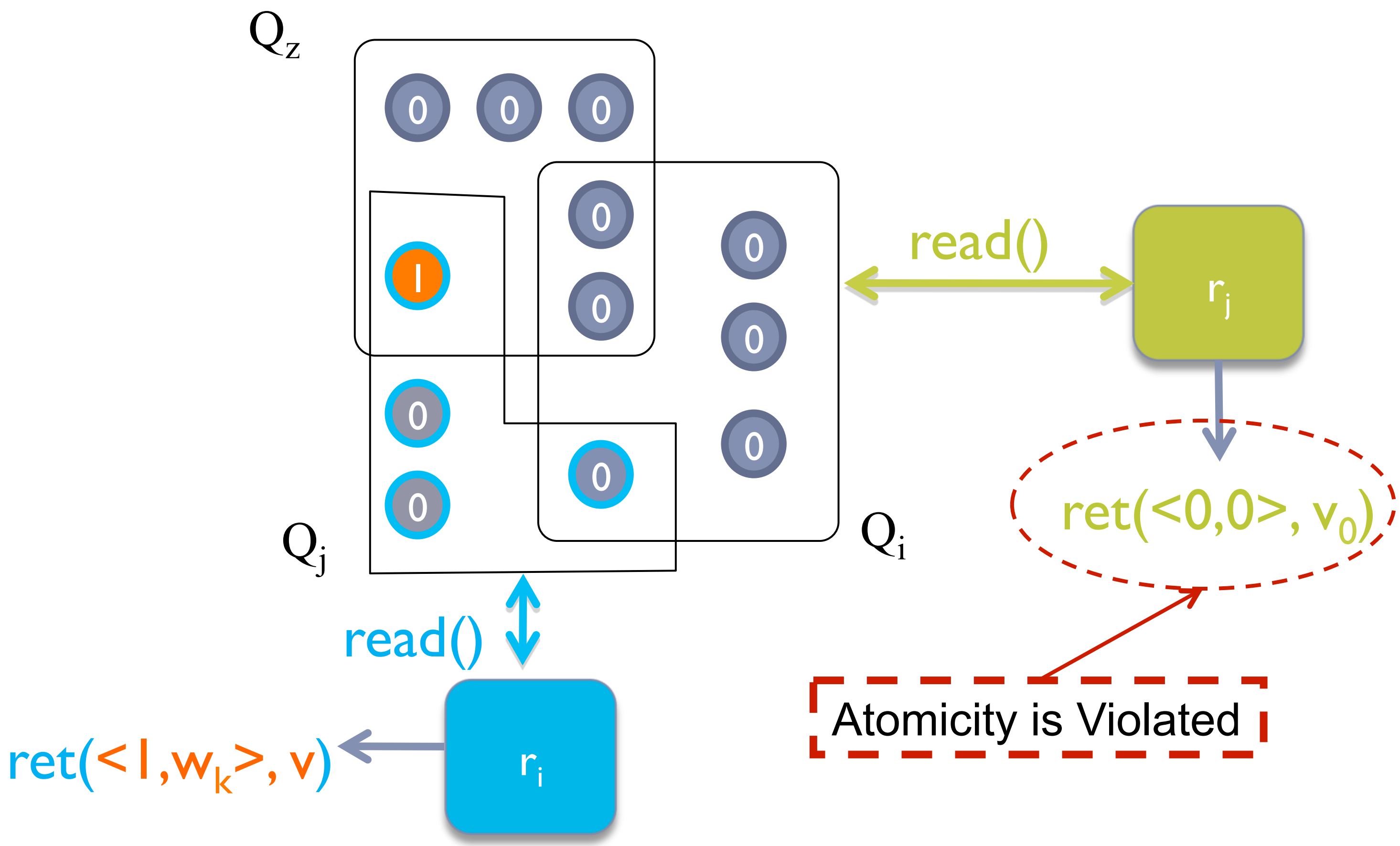
Why a read performs 2 rounds?

Consider the following executions with single round reads:



Why a read performs 2 rounds? (Cont.)

Extend execution Ex(b) with a read from rj:



Folklore Belief: **Reads must Write in MR environments**

Question

Can we allow reads and writes to be fast (single round) and still guarantee atomicity?

Answer: **YES!!**

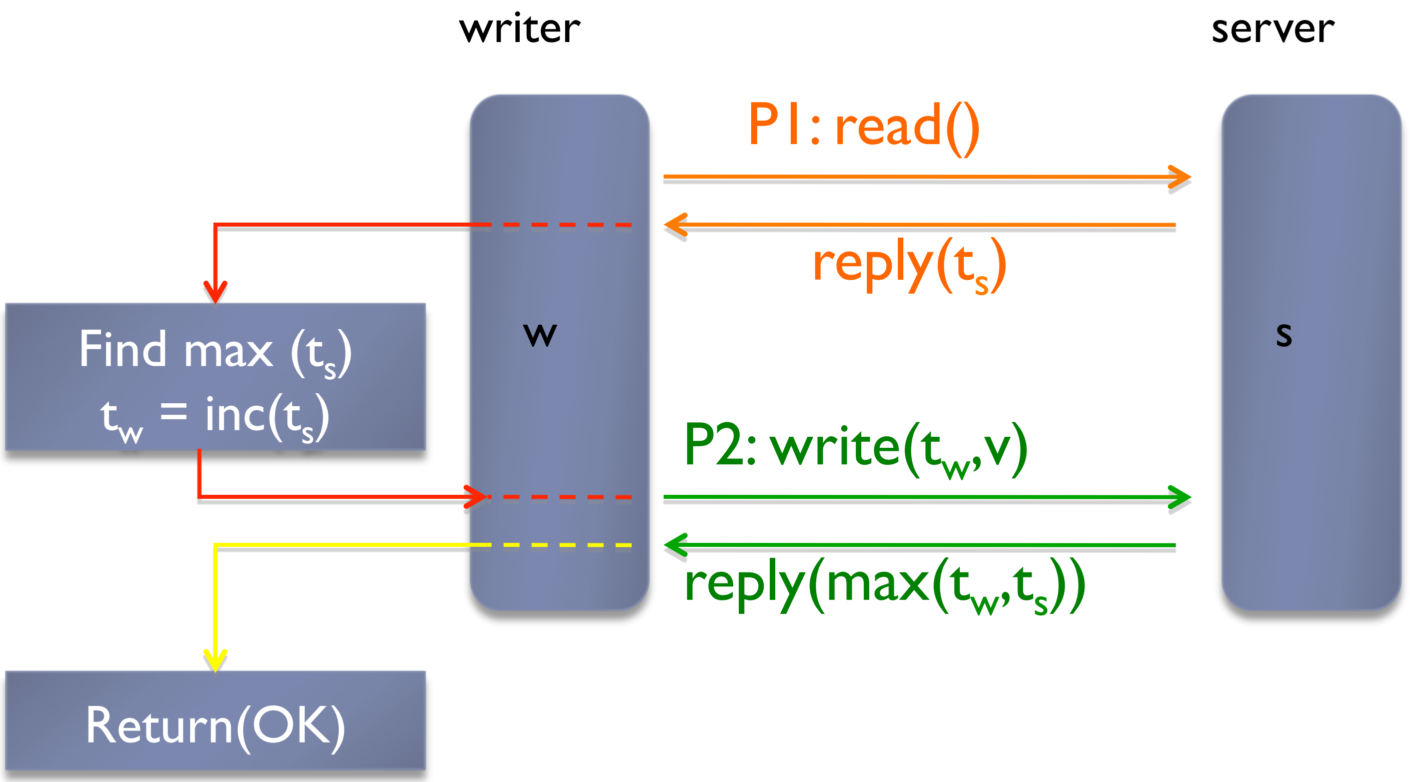
New Technique - SSO

[Englert et. al 09]

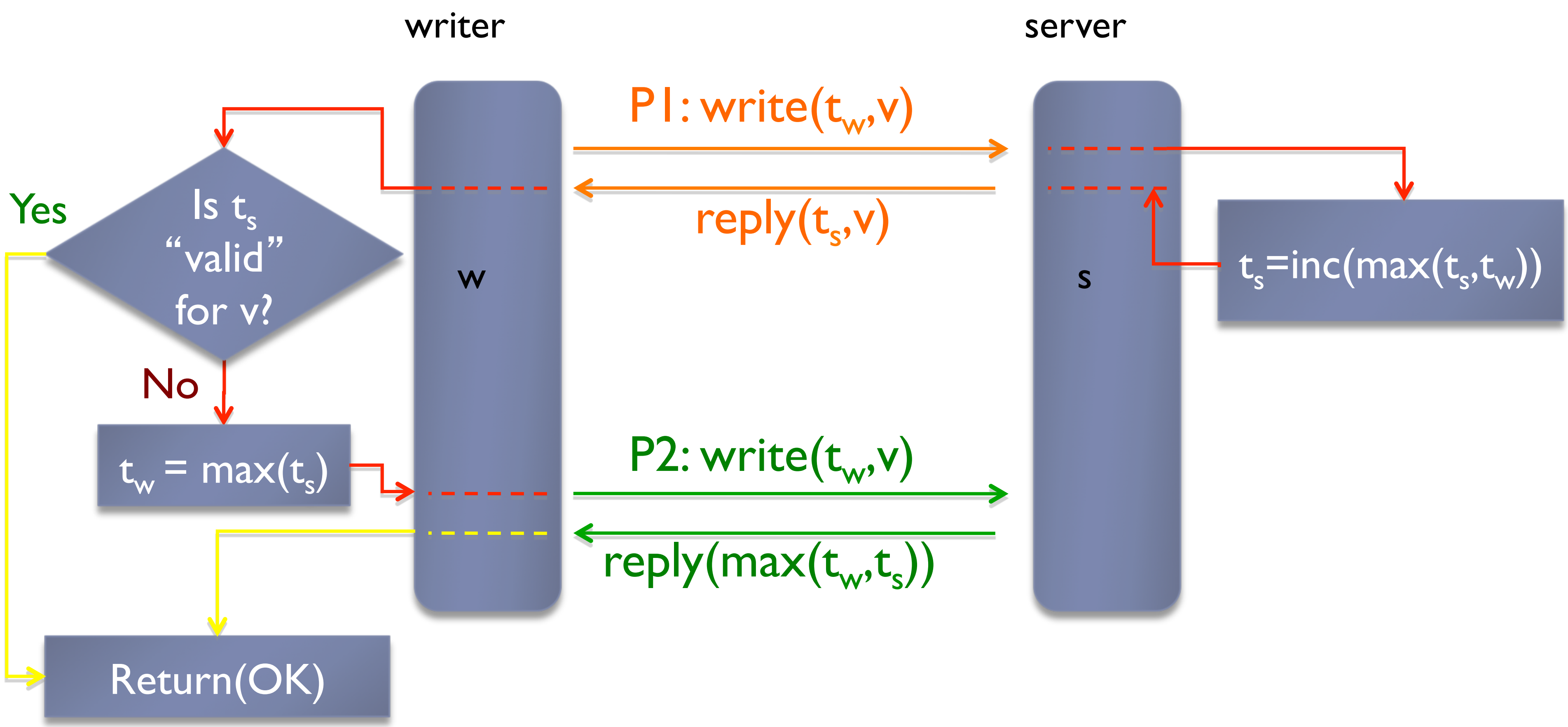
- SSO: Server Side Ordering
  - Tag is incremented by the servers and not by the writer.
    - Generated tags may be different across servers
    - Clients decide operation ordering based on server responses

- SSO Algorithm
  - Enables Fast Writes and Reads -- first such algorithm
  - Allows Unbounded Participation

Traditional Writer-Server Interaction



SFW Writer-Server Interaction



Algorithm: SFW (in a glance)

Write Protocol: one or two rounds

- P1: Collect candidate tags from a quorum
  - Exists tag t propagated in a bigger than (n/2-1)-wise intersection (PREDICATE PW)
    - YES – assign t to the written value and return => **FAST**
    - NO - propagate the unique largest tag to a quorum => **SLOW**

Read Protocol: one or two rounds

- P1: collect list of writes and their tags from a quorum
  - Exists max write tag t in a bigger than (n/2-2)-wise intersection (PREDICATE PR)
    - YES – return the value written by that write => **FAST**
    - NO – is there a confirmed tag propagated to (n-1)-wise intersection => **FAST**
    - NO - propagate the largest confirmed tag to a quorum => **SLOW**

Server Protocol

- Increment tag when receive write request and send to read/write the latest writes

Predicates: Read and Write

Writer predicate for a write ω **(PW):** $\exists\ \tau, \mathbb{Q}^i, MS$ where: $\tau \in \{\langle ., \omega\rangle : \langle ., \omega\rangle \in m(\omega)_{s,w}.inprogress \ \wedge\ s \in Q\}$, $MS = \{s : s \in Q \ \wedge\ \tau \in m(\omega)_{s,w}.inprogress\}$, and $\mathbb{Q}^i \subseteq \mathbb{Q}, 0 \leq i \leq \lfloor \frac{n}{2} - 1 \rfloor$, s.t. $(\bigcap_{\mathcal{Q} \in \mathbb{Q}^i \cup \{Q\}} \mathcal{Q}) \subseteq MS$.

Reader predicate for a read ρ **(PR):** $\exists\ \tau, \mathbb{Q}^j, MS$, where: $\max(\tau) \in \bigcup_{s \in Q} m(\rho)_{s,r}.inprogress$, $MS = \{s : s \in Q \ \wedge\ \tau \in m(\rho)_{s,r}.inprogress\}$, and $\mathbb{Q}^j \subseteq \mathbb{Q}, 0 \leq j \leq \lfloor \frac{n}{2} - 2 \rfloor$, s.t. $(\bigcap_{\mathcal{Q} \in \mathbb{Q}^j \cup \{Q\}} \mathcal{Q}) \subseteq MS$.

Lower bounds

<u>Theorem</u>: No execution of safe register implementation that use an N-wise quorum system, contains more than $N-1$ consecutive, quorum shifting, fast writes.

<u>Theorem</u>: It is impossible to get MWMR safe register implementations that exploit an N-wise quorum system, if

$$|W \cup R| > N-1$$

Remarks

Remark: SSO algorithm is near optimal since it allows up to $\left(\frac{N}{2}-1\right)$ consecutive, quorum shifting, fast writes.

The Weak Side of SFW

- Predicates are Computationally Hard
  - NP-Complete
- Restriction on the Quorum System
  - Deploys n-wise Quorum Systems
  - Guarantees fastness iff n>3

The Good News…

- Approximation Algorithm (APRX-SFW)
  - Polynomial
  - Log-approximation
    - log|S| times the optimal number of fast operations

- Algorithm CWFR
  - Based on Quorum Views
    - SWMR prediction tools
  - Fast operations in General Quorum Systems
  - Trades Speed of Write operations
    - Two Round Writes

NP-Completeness

K-SET-INTERSECTION: (captures both PR and PW)

Given a set of elements U, a subset of those elements $M \subseteq U$, a set of subsets $\mathbb{Q} = \{Q_1, \ldots, Q_n\}$ s.t. $Q_i \subseteq U$, and an integer $k \leq |\mathbb{Q}|$, a set $I \subseteq \mathbb{Q}$ is a k-intersecting set if: $|I| = k$, $\bigcap_{Q \in I} Q \subseteq M$, and $\bigcap_{Q \in I} Q \neq \emptyset$.

Theorem: K-SET-INTERSECTION is NP-complete (reduction from 3-SAT).

k-Set-Intersection Approximation

- Greedy algorithm
  - Uses Set Cover greedy approximation algorithm at its core

K-SET-COVER:

Given a universe U of elements, a collection of subsets of U, $S = \{S_1, \ldots, S_z\}$, and a number k, find at most k sets of S such that their union covers all elements in U.

k-Set-Intersection Approximation

- Given $(U, M, \mathbb{Q}, k)$ do:

Step 1:

$$\forall m \in M, T_m = \{(U \setminus M) \setminus (Q_i \setminus M) : m \in Q_i\}$$

Step 2: Run k-SET-COVER greedy algorithm on $(U \setminus M, T_m, k)$

- 2a: Pick $R \in T_m$ with the maximum uncovered elements
- 2b: Take the union of every set picked in 2a
- 2c: If the union is $U \setminus M$ go to step 3, else if we picked less than k sets go to 2a, else repeat for another $m \in M$

Step 3:

- For every set $(U \setminus M) \setminus (Q_i \setminus M)$ in the set cover, add Q_i in the intersecting set

Algorithm Rationale

- Let for $m \in M, Q_i$

$$R_{m,i} \in T_m : R_{m,i} = (U \setminus M) \setminus (Q_i \setminus M)$$

- If we can find k sets such that:

$$R_{m,1} \cup \ldots \cup R_{m,k} = U \setminus M$$

- By de Morgan's: $\overline{R_{m,1}} \cap \ldots \cap \overline{R_{m,k}} = \emptyset$

- Since $\overline{R_{m,i}} = (Q_i \setminus M)$ and $m \in Q_i$ for $i \in [1, \ldots, k]$

$$m \in Q_1 \cap \ldots \cap Q_k \text{ and } Q_1 \cap \ldots \cap Q_k \subseteq M$$

Approximation Algorithm: APRX-SFW

- Adopt k-Set-Intersection Approximation:
  - $U = S$ the set of servers
  - $\mathbb{Q} = \{Q_1, \ldots, Q_q\}, Q_i \subset S$ is the quorum system
  - $M \subseteq S$ the servers that replied with the latest value
  - k the number of quorums required by the predicates

- Log-Approximation
  - Invalidates RP and WP a factor of log|S| times

- What does it mean for SFW?
  - Extra Communication Rounds (esp. for writes)
  - Slower acceptance of a new value
  - Does not affect correctness

Unrestricting Quorums

- APRX-SFW
  - Improves Computation Time
  - Still relies on n-wise Quorum Systems
    - n>3 to allow fast operations

Can we allow fast operations in the MWMR when deploying General Quorum Systems?

Answer: **YES!!**

Tool: Quorum Views

Used in the SWMR [Georgiou et al. 08]

Idea:

- Try to determine the state of the write operation based on the distribution of the latest value in the replied quorum.

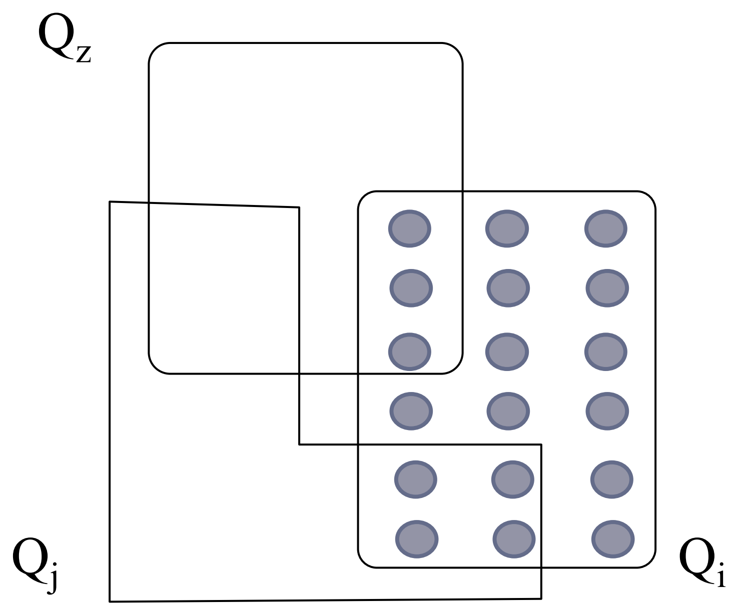
- Write State in the First Round of Read Operation

  Determinable => Read is Fast

  Undeterminable => Read is Slow

Determinable Write - Qview(1)

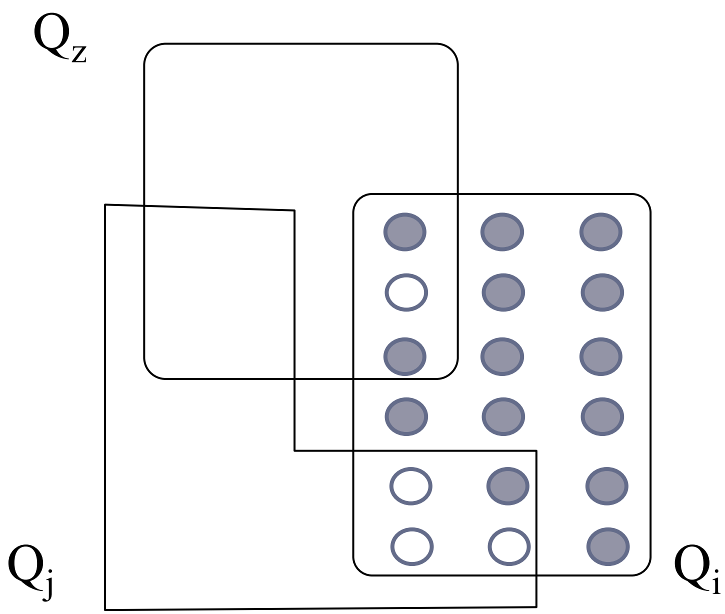
- All members of a quorum contain maxTag



(Potentially) Write Completed

Determinable Write - Qview(2)

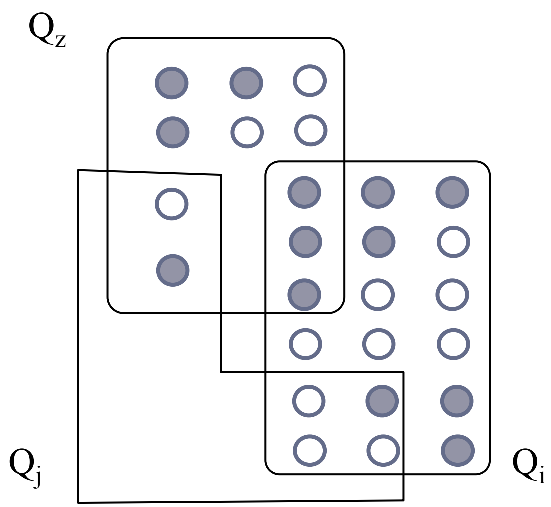
- Every intersection contains a member with tag<maxTag



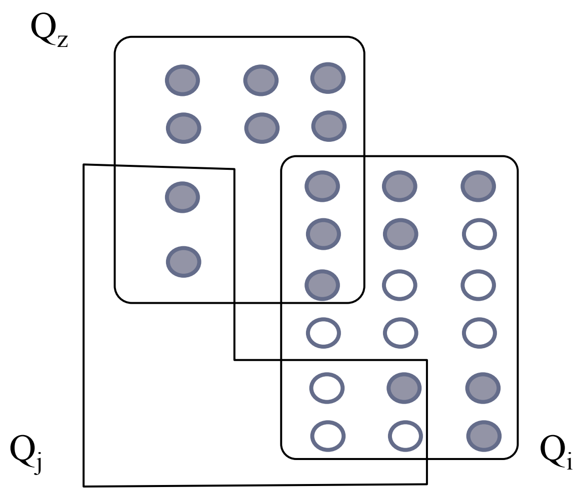
(Definitely) Write <maxTag,v> Incomplete

Undeterminable Write - Qview(3)

- There is intersection with all its members with tag=maxTag



qV(3) and Incomplete Write

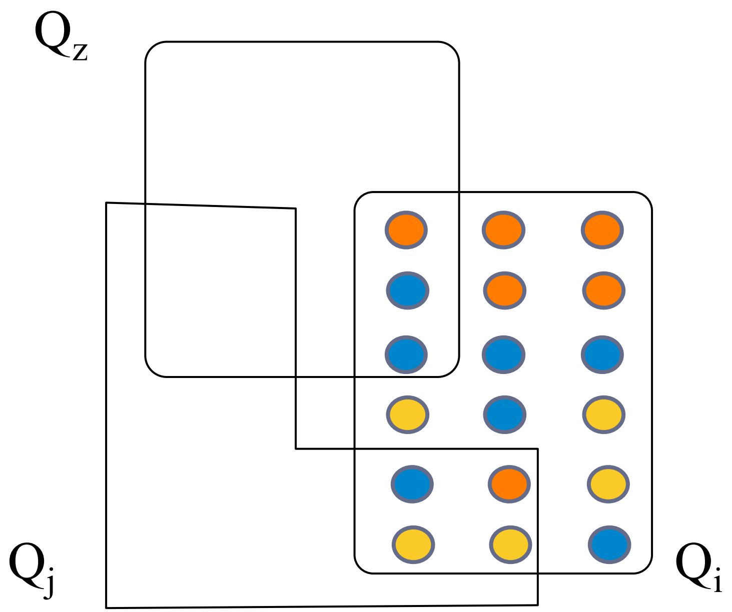


qV(3) and Complete Write

Undeterminable => second Com. Round

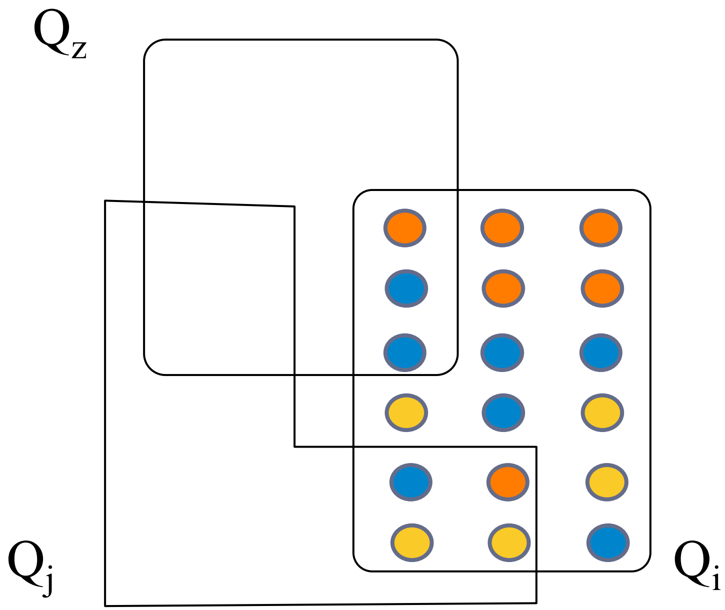
What happens in MWMR?

- MWMR environment
  - Concurrent writes
  - Multiple concurrent values
- For values <tag1,v1> , <tag2, v2>, <tag3,v3>
  - Let tag1 < tag2 < tag3



Idea: Uncover the Past

- Discover the latest potentially completed write
- For values <tag1,v1> , <tag2, v2>, <tag3,v3>:
  - <tag3,v3> not completed (servers possibly contained <tag2, v2>)
  - <tag2, v2> not completed (servers possibly contained <tag1,v1>)
  - <tag1,v1> potentially completed



Algorithm: CWFR

Traditional Write Protocol: two rounds

- P1: Query a single quorum for the latest tag
- P2: Increment the max tag, send <newtag, v> quorum

Read Protocol: one or two rounds

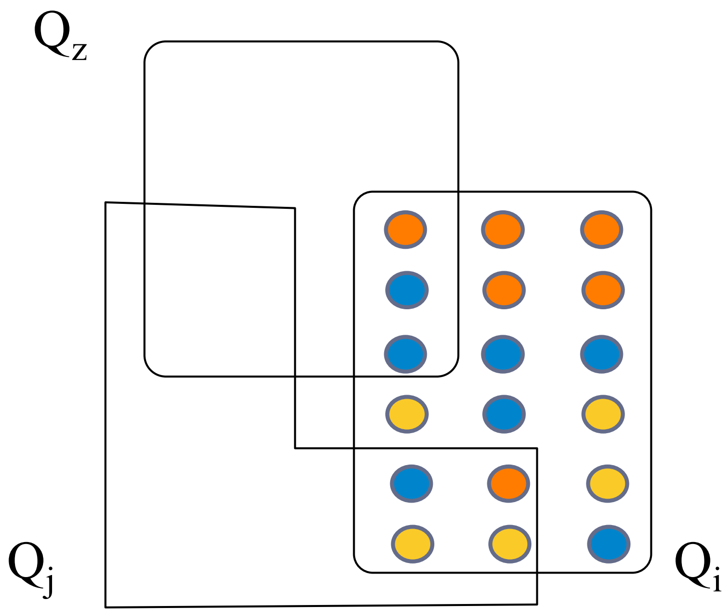
- Iterate to discover smallest completed write
- P1: receive replies from a quorum Q
  - $QView_Q(1)$ – **Fast**: return maxTag of current iteration
  - $QView_Q(2)$ – remove servers with maxTag and re-evaluate
  - $QView_Q(3)$ – **Slow**: propagate and return $maxTag_0$

Server Protocol: passive role

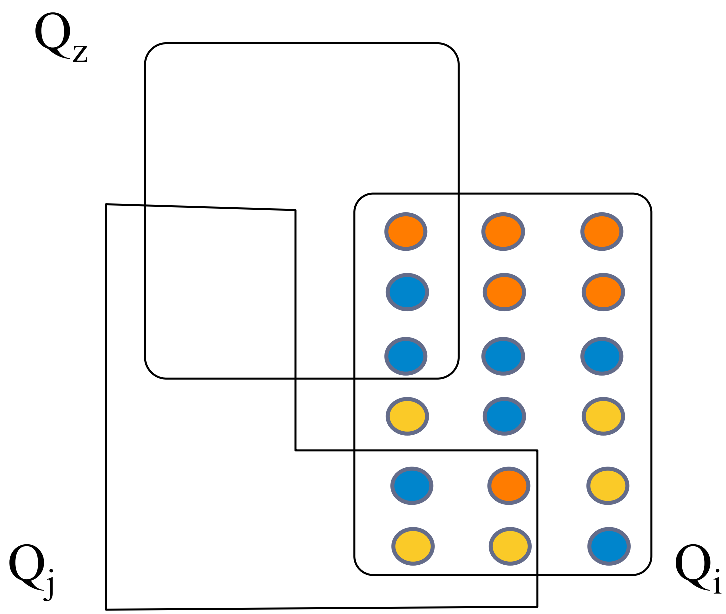
- Receive requests, update local timestamp and return <tag,v>

Read Iteration: Discard Incomplete Tags

- For values <tag1,v1> , <tag2, v2>, <tag3,v3>:
  - <tag3,v3> not completed: remove servers that contain <tag3,v3>
  - <tag2, v2> not completed: remove servers that contain <tag2, v2>
  - <tag1,v1> potentially completed in Q_i
    - Qview(1) : all remaining servers contain <tag1,v1>



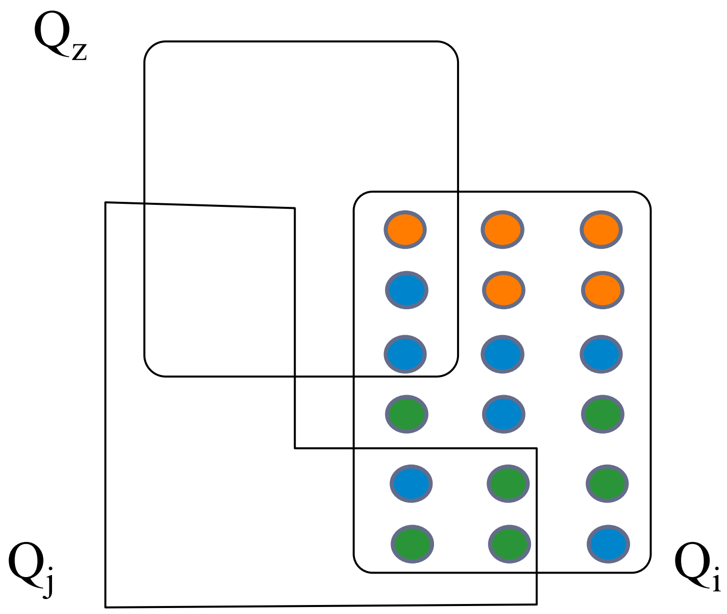
Server Removal



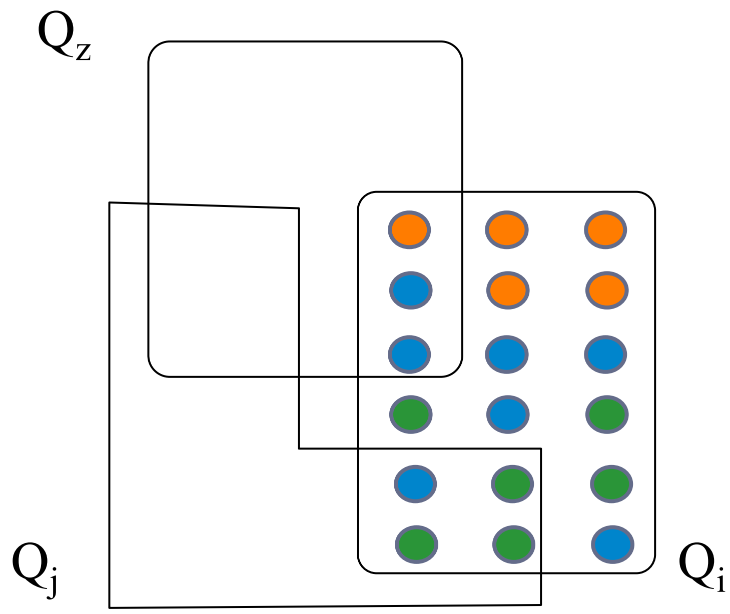
Past Prediction

Read Iteration: Discard Incomplete Tags

- For values <tag1,v1> , <tag2, v2>, <tag3,v3>:
  - <tag3,v3> not completed: remove servers that contain <tag3,v3>
  - <tag2, v2> potentially completed in Q_j
    - Qview(3) : an intersection of the remaining servers contains <tag2, v2>
    - P2: propagate <tag3,v3> to a complete quorum (help <tag3,v3> to complete)

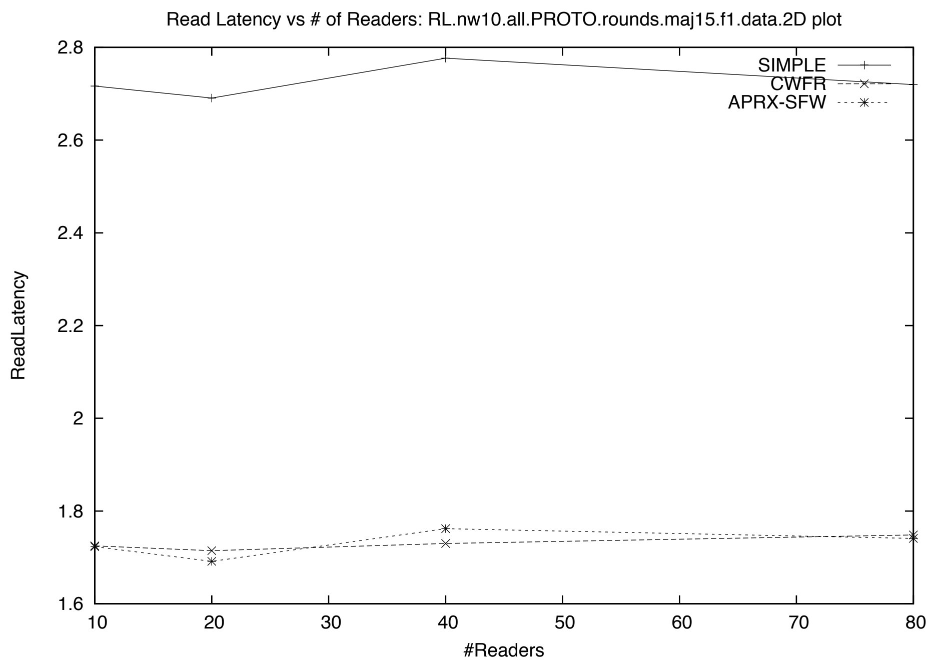


Server Removal



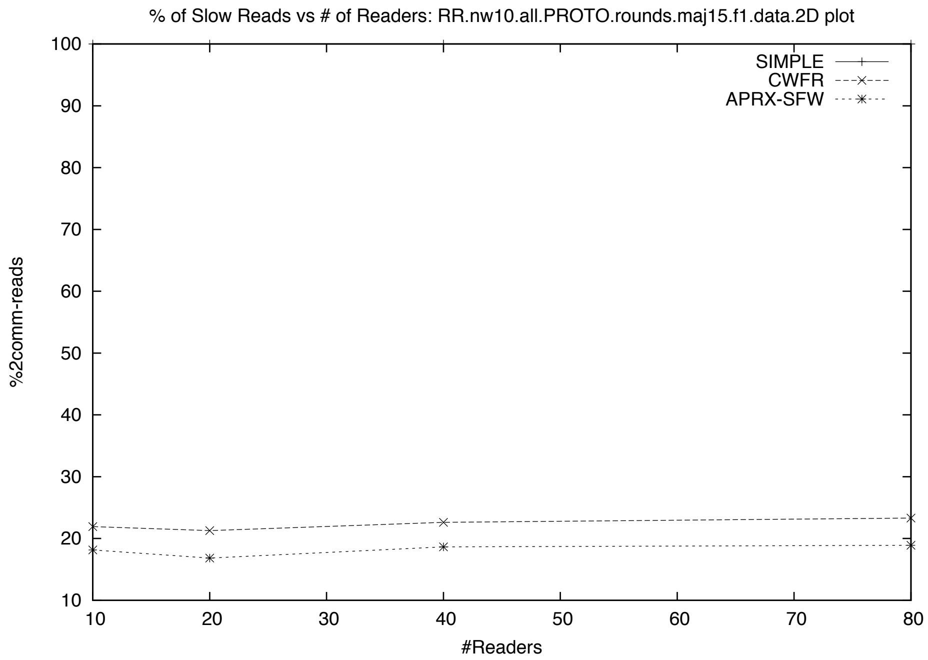
Past Prediction

APRX-SFW – CWFR: NS2 Simulation



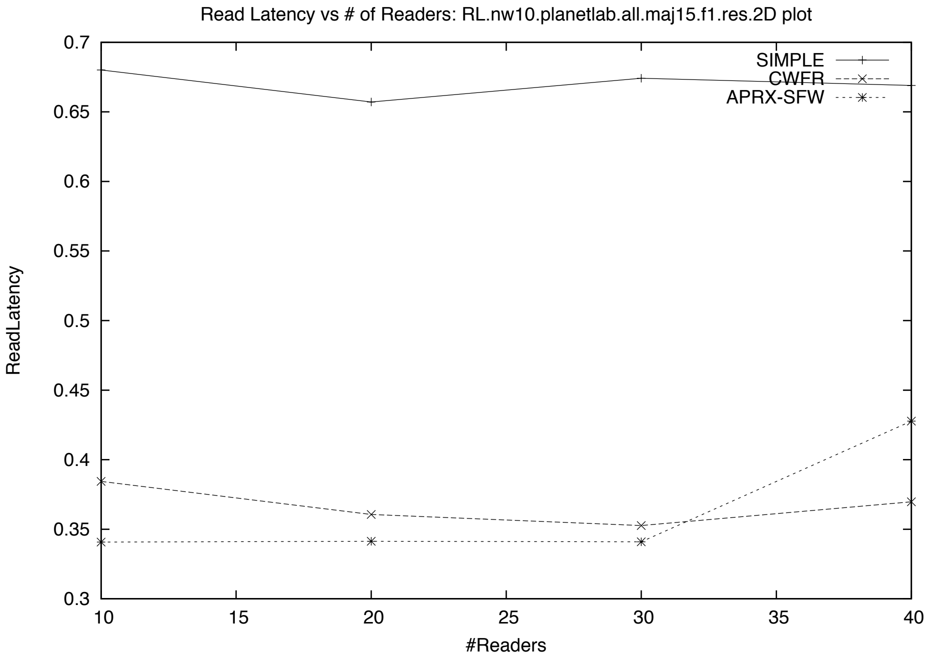
Latency

Rounds



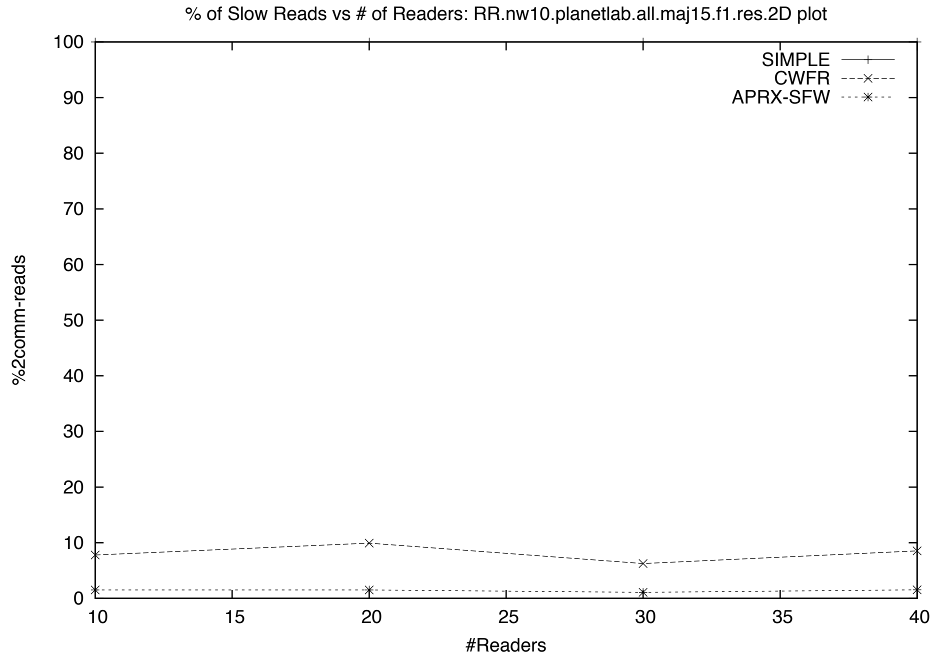
14-wise Quorum System

APRX-SFW – CWFR: Planetlab



Latency

Rounds



14-wise Quorum System

Conclusions

- Presented two Atomic Register MWMR implementations
  - Computation and Communication factor

- Algorithm: APRX-SFW
  - Polynomial-Approximation of SFW predicates
  - log|S|-approximation
  - Requires n-wise Quorum Systems for n>3

- Algorithm: CWFR
  - General Quorum systems
  - Trades the Speed of write operations

- Experiments on NS2 and Planetlab
  - Both algorithms overperform classic approach
  - Bigger Intersections favor the APRX-SFW