institute IMdea networks

NAZARBAYEV UNIVERSITY

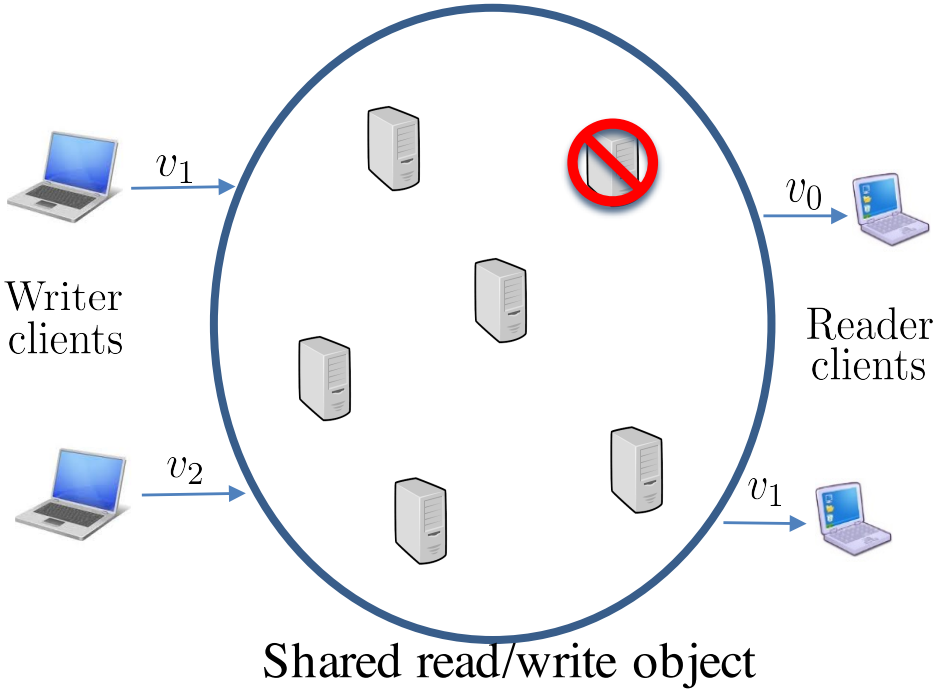
Making "Fast" Atomic Operations Computationally Tractable

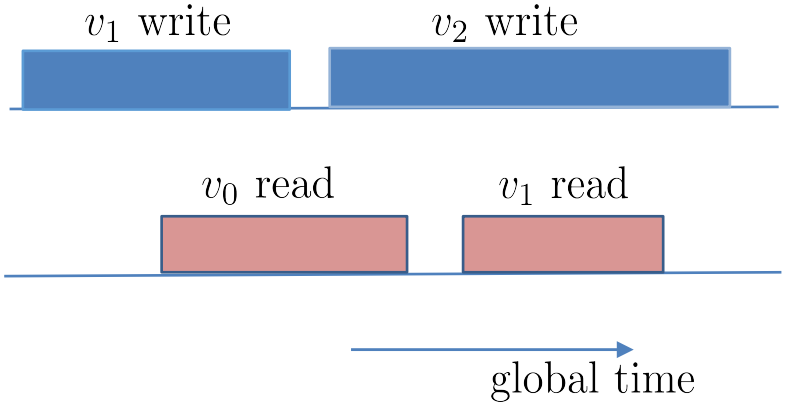
Antonio Fernández Anta, **Nicolas Nicolaou**, and Alexandru Popa

Developing the Science of Networks

Nicolas Nicolaou

Problem Statement





Implementing a fault-tolerant shared object in an asynchronous, message-passing environment:

- Availability + Survivability => use redundancy
- Asynchrony + Redundancy => concurrent operations
- Behavior of concurrent operations => consistency semantics
  - Safety, Regularity, Atomicity [Lamport86]

System Model: Definitions

Components

- Clients: 1 writer & R readers (SWMR)
- Servers: S replica hosts

Operations

- write(*v*): updates the object value to *v*
- read(): retrieves the object value
- Well-Formedness (only a single operation at a time)

Communication

- Asynchronous
- Message-Passing
- Reliable Channels (messages are not lost or altered)

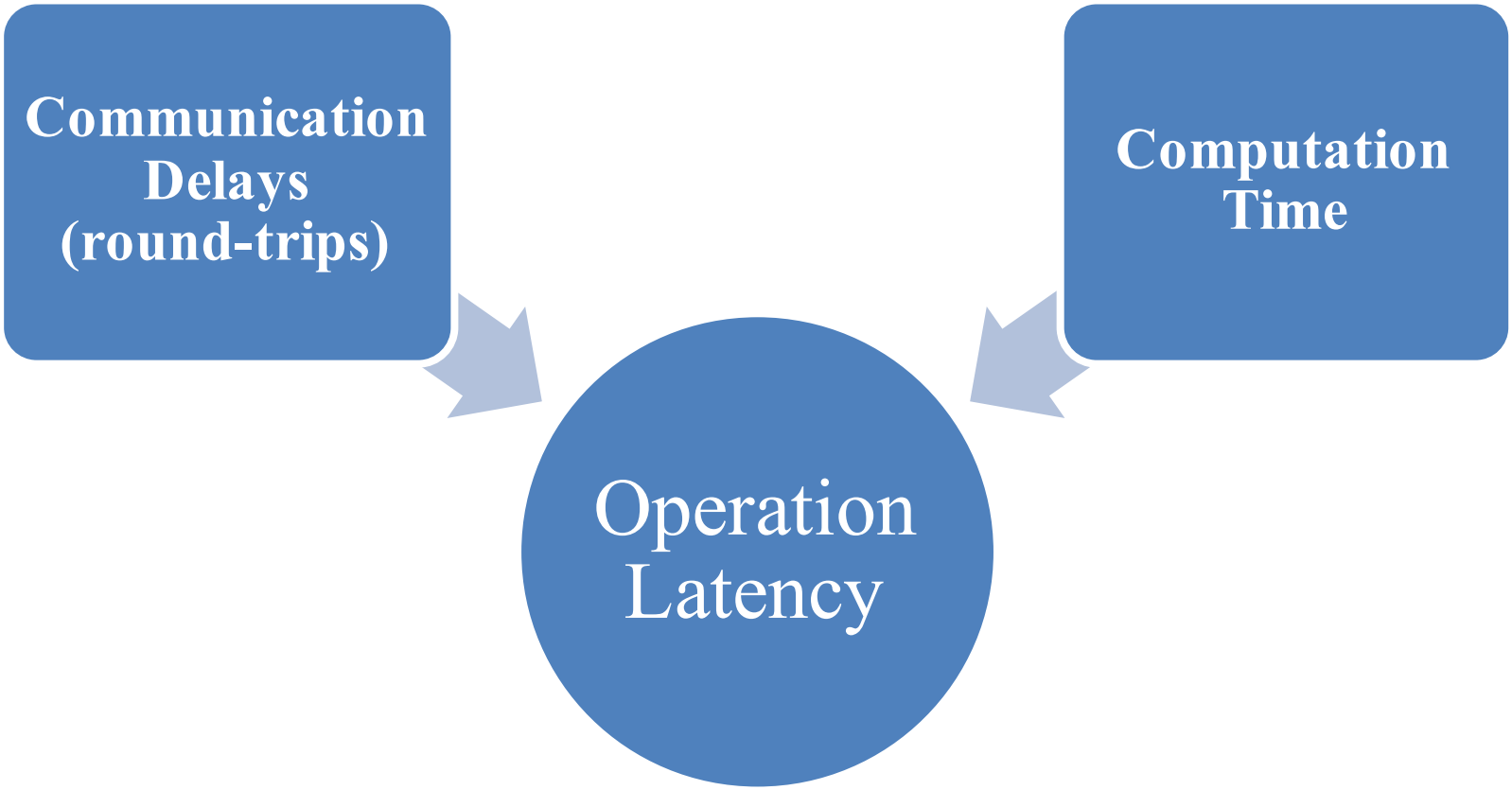
Failures

- Crashes
- Any reader or the writer
- Up to minority of servers

Consistency Model: Atomicity/Linearizability

- Provides the illusion that operations happen in a sequential order
  - a read returns the value of the preceding write
  - a read returns a value at least as recent as that returned by any preceding read

Complexity Measure



Algorithm ABD: Recalling the past

- [Attiya, Bar-Noy, Dolev 1996] (Dijkstra Prize 2011)
- Order Operations by using <ts, v> pairs.

Reader Protocol (2 phases)

- Phase 1:
  - Send read to all
  - Collect <ts,v> from a majority
  - Discover max(<ts,v>)
- Phase 2:
  - Send max(<ts,v>) to all
  - Collect ack from a majority and return v

Writer Protocol

- ts++ //increment ts
- Send <ts,v> to all
- Wait for a majority to reply

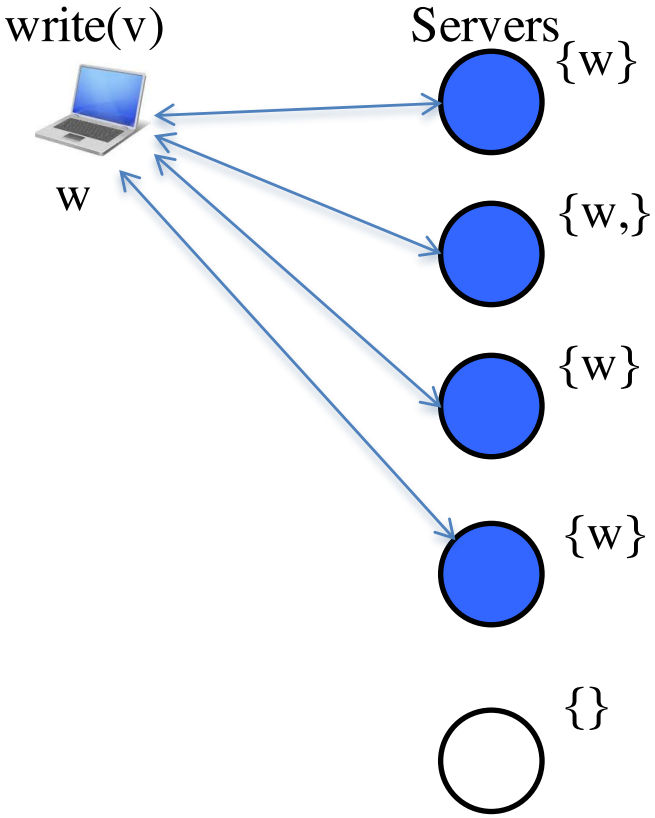
Server Protocol (Upon rcv a msg m from i)

- if m.ts > ts
  - Update local <ts,v>
- Send local <ts,v> to i

Reads must Write!
(2 round-trips)

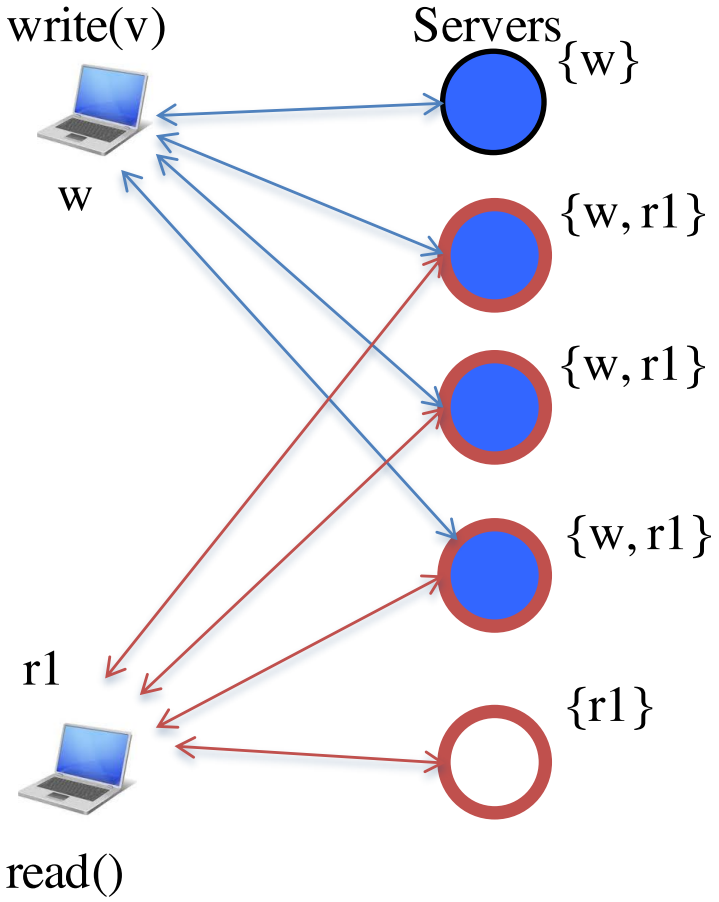
Algorithm FAST: Reducing communication

- [Dutta et al. 2004, 2010] made a nice observation
- Consider f=1, S=5 and let operation to communicate with S-f servers instead of majority



Algorithm FAST: Reducing communication

- [Dutta et al. 2004, 2010] made a nice observation
- Consider f=1, S=5 and let operation to communicate with S-f servers instead of majority



After 1 read

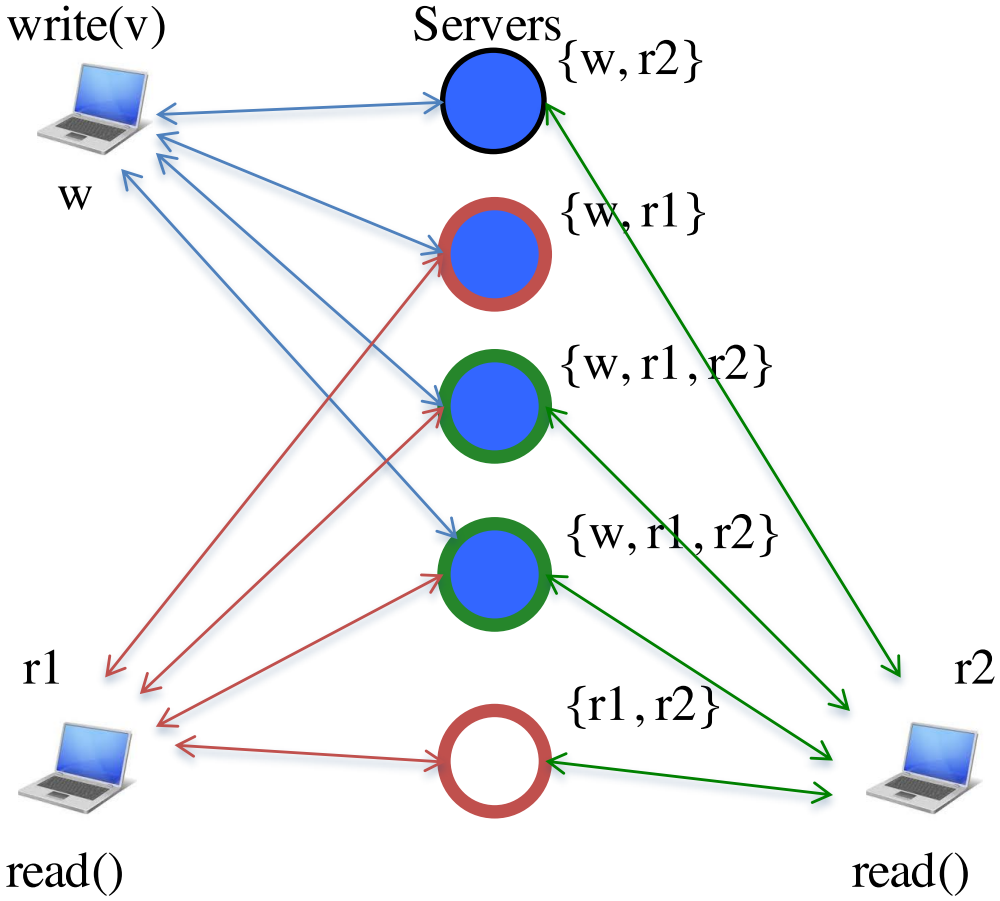
- At least S-2f servers
- Each replies to both {w, r1}

Nicolas Nicolaou

institute IMdea networks

Algorithm FAST: Reducing communication

- [Dutta et al. 2004, 2010] made a nice observation
- Consider f=1, S=5 and let operation to communicate with S-f servers instead of majority



After 1 read

- At least S-2f servers
- Each replies to both {w, r1}

After 2 reads

- At least S-3f servers
- Each replies to all {w, r1, r2}

Nicolas Nicolaou

institute IMdea networks

Algorithm FAST: Single round-trip operations

- Constructed a predicate that allowed all writes and reads to complete in a single round-trip

$$\exists \alpha \in [1, R+1] \wedge MS \subseteq S \text{ s.t.}$$

$$\forall s \in MS, s.ts = maxTS \wedge |MS| \geq S - \alpha f \wedge |\bigcap_{s \in MS} s.seen| \geq \alpha$$

Server Protocol (Upon rcv a msg m from i)

- if m.ts > ts
  - Update local info
  - Reset seen set to {i}
- else
  - Add i in seen set
- Send ts and seen set to i

Writer Protocol (1 phase)

(Same as ABD)

Reader Protocol (1 phase)

- Send read to all
- Collect <ts,v> and seen sets from S-f
- Discover maxTS = max(ts)
- If predicate is true:
  - return maxTS
- else
  - return maxTS-1

Implications of FAST

- On the number of readers:

$$R < \frac{S}{f} - 2$$

- On the number of writers
  - Impossible in the MWMR model

- What about the computation?

How hard is it to compute the predicate?

Problem 1 (Predicate Formalization)

Input

- Two sets

$$U_1 = \{s_1, \ldots, s_n\}, U_2 = \{p_1, \ldots, p_k\} \quad \text{s.t.} \quad \forall s_i \in U_1, s_i \subseteq U_2$$

- Two integers $\alpha, f \quad \text{s.t.} \quad n - \alpha f \geq 1$

Output

- Is there a set

$$M \subseteq U_1 \quad \text{s.t.} \quad \left| \bigcap_{s \in M} s \right| \geq \alpha \quad \text{and} \quad |M| > n - \alpha f$$

Nicolas Nicolaou

institute IMDEA networks

Nicolas Nicolaou

institute IMdea networks

Problem equivalence to the predicate

$$\exists \alpha \in [1, R] \wedge MS \subseteq S \text{ s.t.}$$

$$\forall s \in MS, s.ts = maxTS \wedge |MS| \geq S - \alpha f \wedge |\bigcap_{s \in MS} s.seen| \geq \alpha$$

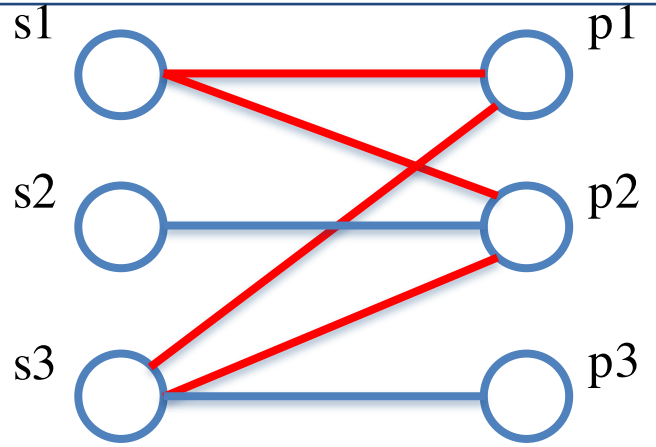
- U_1 : the set of all the seen sets collected
- U_2 : set of reader and the writer identifiers
- M : MS in the predicate
- α, f : the respective α and f in the predicate

Maximum Biclique Problem (MBP)

Definition MBP:

Input

- A bipartite graph $G = (X, Y, E)$
- A positive integer c



A = {s1,s3}, B={p1,p2}
c = 4

Output

- Are there two sets

$$A \subseteq X, B \subseteq Y \quad \text{s.t.} \quad \forall a \in A, \forall b \in B, (a, b) \in E \text{ and } |E| = |A| * |B| \geq c$$

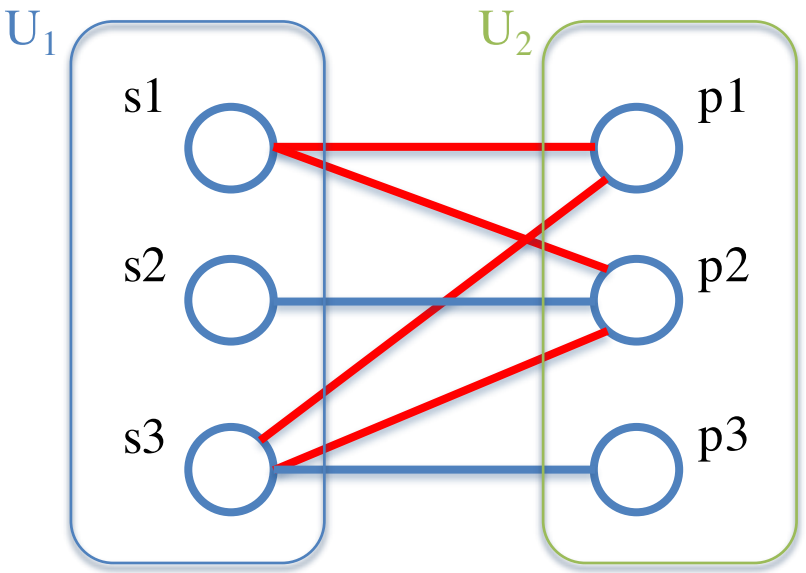
MBP is NP-complete [Peeters 01]

Nicolas Nicolaou

institute IMdea networks

Problem 1 is NP-Hard - Reduction Idea

Input Transformation



- Given the graph $G = (X, Y, E)$
  - Set $U_1 = X$
  - Set $U_2 = Y$
  - $(s_i, p_j) \in E \iff p_j \in s_i$
- Set $c = \alpha(n - \alpha f)$
- Problem 1 returns true if
  - exists $M \subseteq U_1$ and $|M| = n - \alpha f$
  - $\exists P \subseteq U_2, |P| = \alpha$ s.t. $\forall s \in M, \forall p \in P, (s, p) \in E$
- In this case $c = |M| * |P| = \alpha(n - \alpha f)$ and MBP is true

Can we overcome NP-Hardness?

Observation

To avoid the excessive computation we need to avoid the set manipulation.

Question

Can we preserve atomicity if we know how many and not which processes read the latest value?

Algorithm ccFAST: A new predicate

$$\exists \alpha \in [1, R+1] \text{ s.t.}$$
$$MS = \{s : s.ts = maxTS \ \wedge \ s.views \geq \alpha\} \text{ and } |MS| \geq S - \alpha f$$

Server Protocol (Upon rcv a msg m from i)

- if m.ts > ts
  - Update local info
  - Set seen = {i}
- else
  - Add i in seen set
- Send local ts and the size of the seen set to i

Reader Protocol (1 phase)

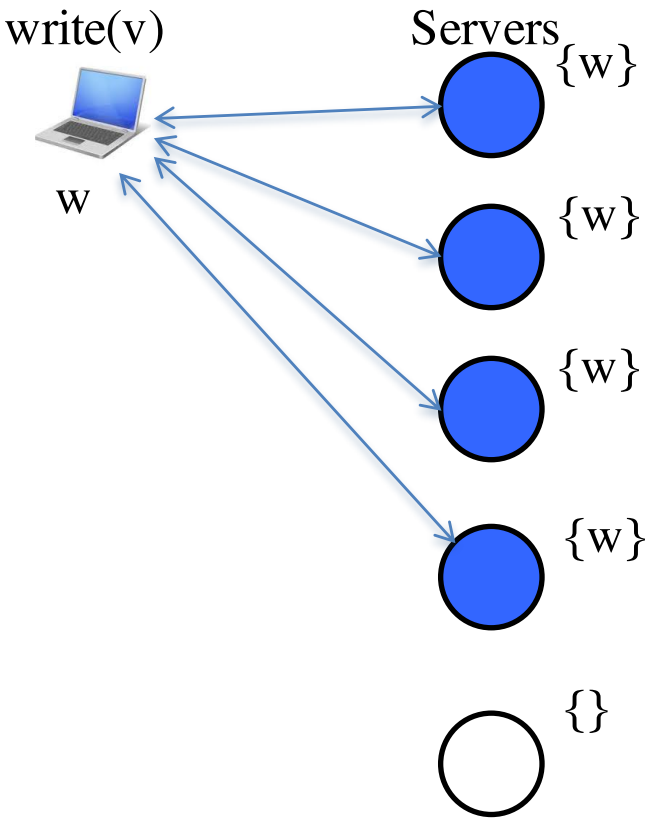
- Send read to all
- Collect <ts,v> and views from S-f
- Discover maxTS = max(ts)
- If predicate is true:
  - return maxTS
- else
  - return maxTS-1

Writer Protocol (1 phase)

(Same as ABD)

Algorithm ccFAST: Idea of the predicate

- Consider f=1, S=5 and let operation to communicate with S-f servers

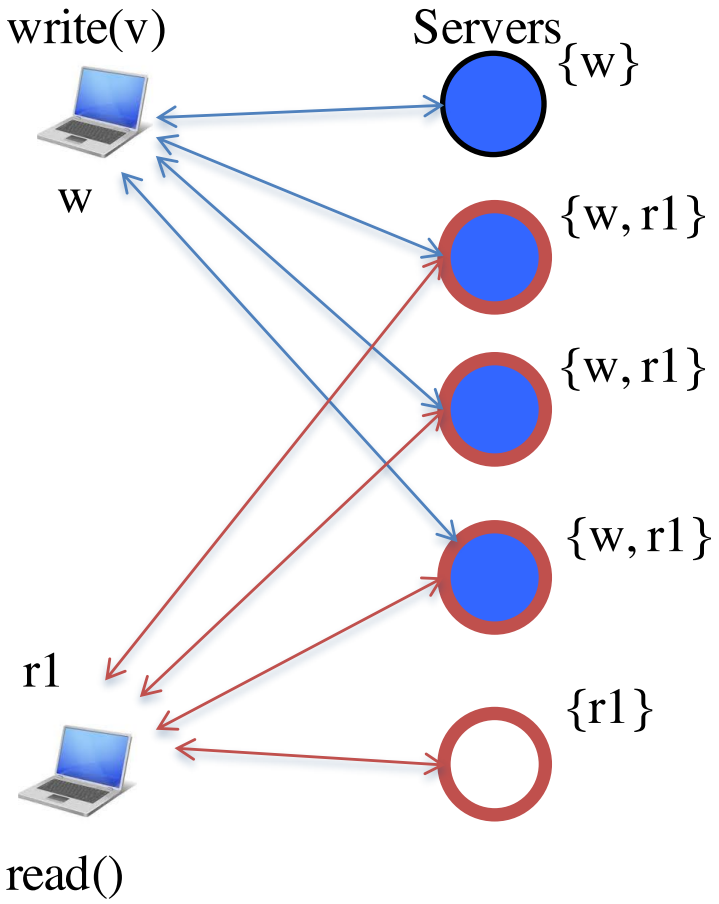


Nicolas Nicolaou

institute IMdea networks

Algorithm ccFAST: Idea of the predicate

- Consider f=1, S=5 and let operation to communicate with S-f servers

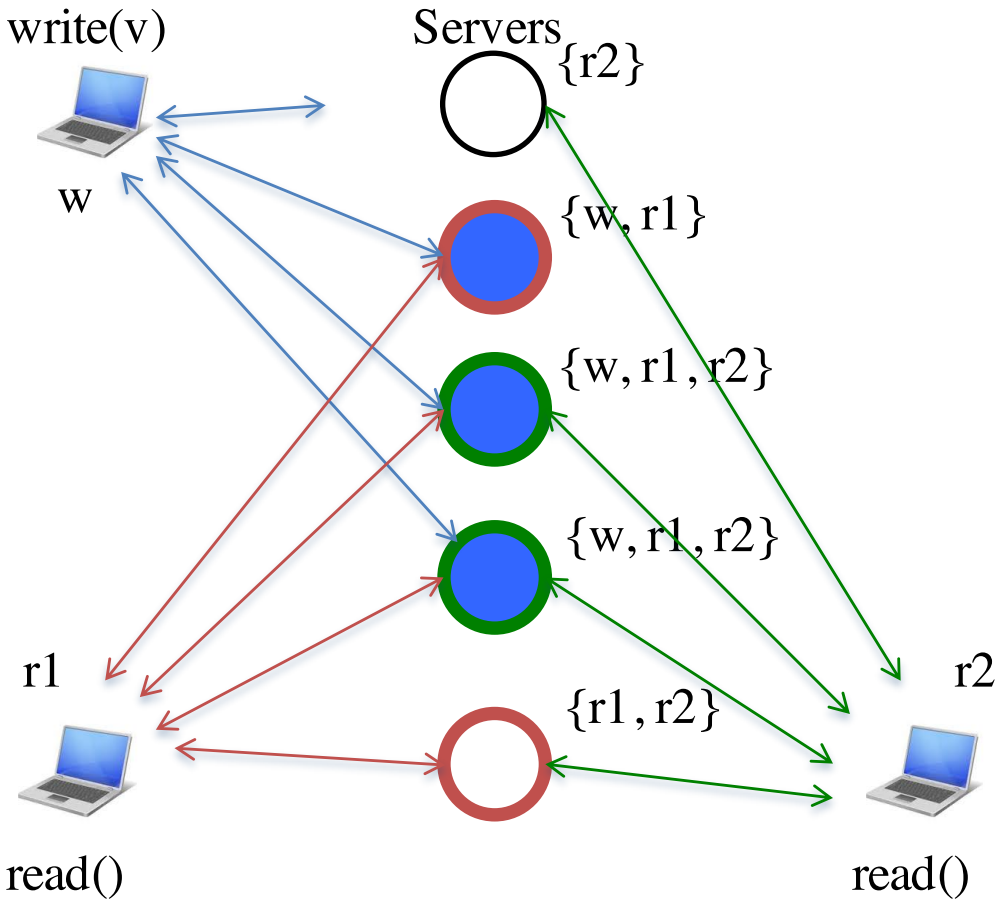


After 1 read

- At least S-2f servers
- Each replies to 2 processes

Algorithm ccFAST: Idea of the predicate

- [Dutta et al. 2004] made a nice observation
- Consider f=1, S=5 and let operation to communicate with S-f servers instead of majority



After 1 read
- At least S-2f servers
- Each replies to 2 processes

After 2 reads
- At least S-3f servers
- Each replies to 3 processes

Algorithm ccFAST: Idea of the predicate

After k-1 reads

- At least S-kf servers
- Each replies to k processes

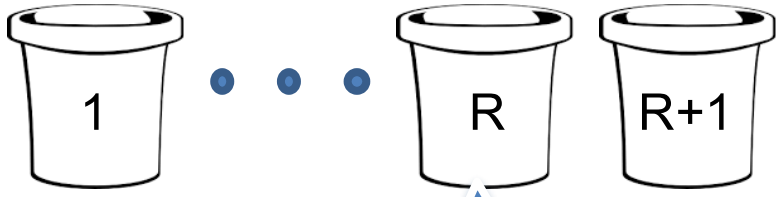
- How big can k be?

$$S - kf > f \Rightarrow k < \frac{S}{f} - 1$$

- Since k is the number of processes then

$$R + 1 < \frac{S}{f} - 1 => R < \frac{S}{f} - 2$$

Computing the new predicate in linear time



Bucket R counts how many servers replied with s.views = R

Linear Complexity: O(S)

Algorithm

- Given b[1…R+1]
- For each s that replied
  - if s.ts = maxTS
    - Increment b[s.views]
- For bucket α in R+1 to 2
  - If b[α] >= S-αf return true
  - Else "empty" b[α] in b[α-1]
    - b[α-1] += b[α]

Complexity Comparison

- Server Messages
  - ABD: [<ts, v>]
  - FAST: [<ts,v>, set seen]
  - ccFAST: [<ts,v>, int views]

Algorithm	WR	RR	WC	RC	WB	RB
ABD	1	2	$O(1)$	$O(\lvert\mathcal{S}\rvert)$	$O(\lg \lvert V\rvert)$	$O(\lg \lvert V\rvert)$
FAST	1	1	$O(1)$	$O(\lvert\mathcal{S}\rvert^2 \cdot 2^{\lvert\mathcal{S}\rvert})$	$O(\lg \lvert V\rvert)$	$\Theta(\lvert\mathcal{S}\rvert + \lg \lvert V\rvert)$
ccFAST	1	1	$O(1)$	$O(\lvert\mathcal{S}\rvert)$	$O(\lg \lvert V\rvert)$	$O(\lg \lvert\mathcal{S}\rvert + \lg \lvert V\rvert)$

WR/RR: write/read round-trips
WC/RC: write/read computation demands
WB/RB: write/read message size in bits

Conclusions

- We showed that FAST is not computationally tractable
- Proposed a new predicate that
  - Preserves "fast" behavior from the operations
  - Reduces the messages sizes
  - Can be computed in linear time
  - Preserves Atomicity
- Presented an algorithm that computes the proposed predicate in linear time

Redefined "fastness" for Atomic Operations

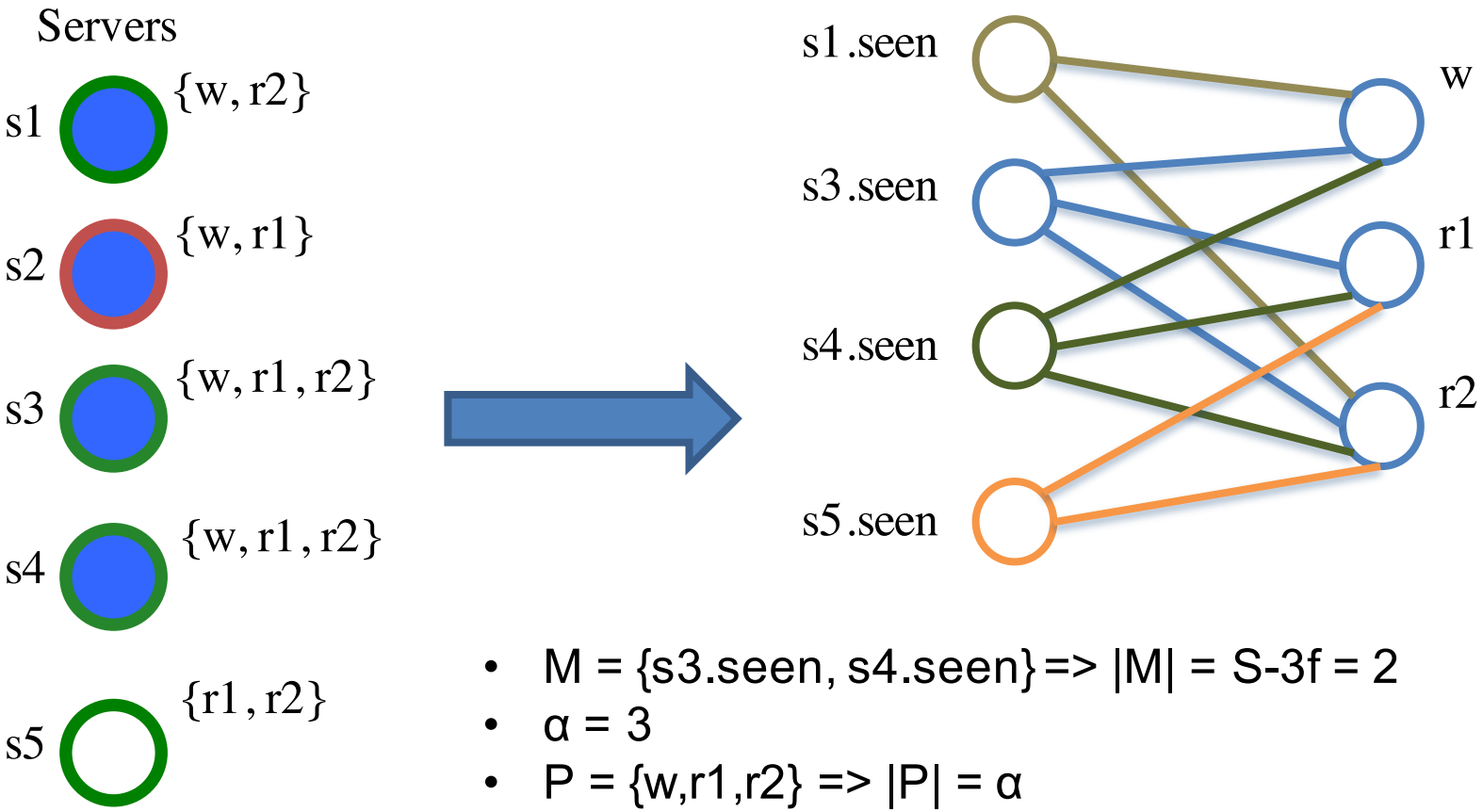
Thank you !

Nicolas Nicolaou

institute IMDEA networks

Reduction Example

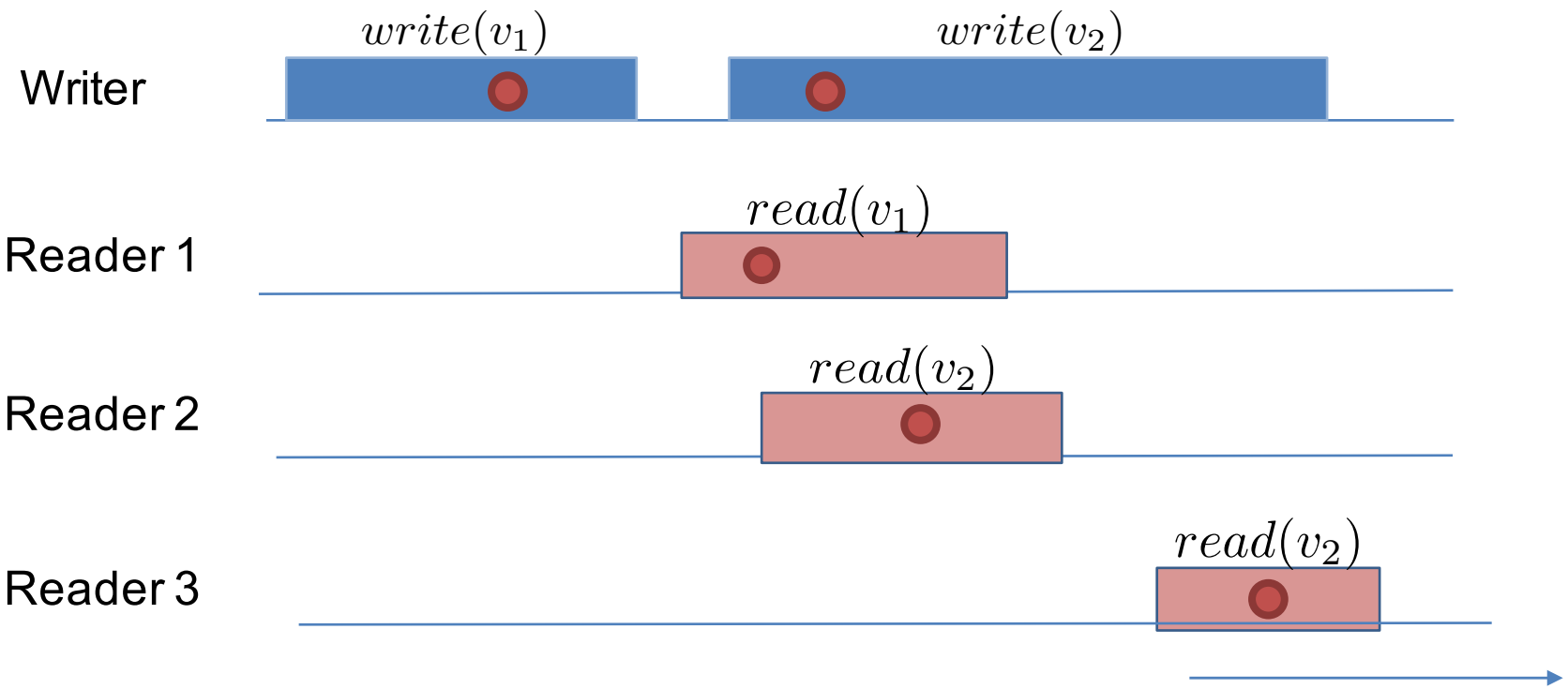
- Consider Previous Example (f=1, S=5)



- M = {s3.seen, s4.seen} => |M| = S-3f = 2
- α = 3
- P = {w,r1,r2} => |P| = α
- c = α (S – αf) = 6

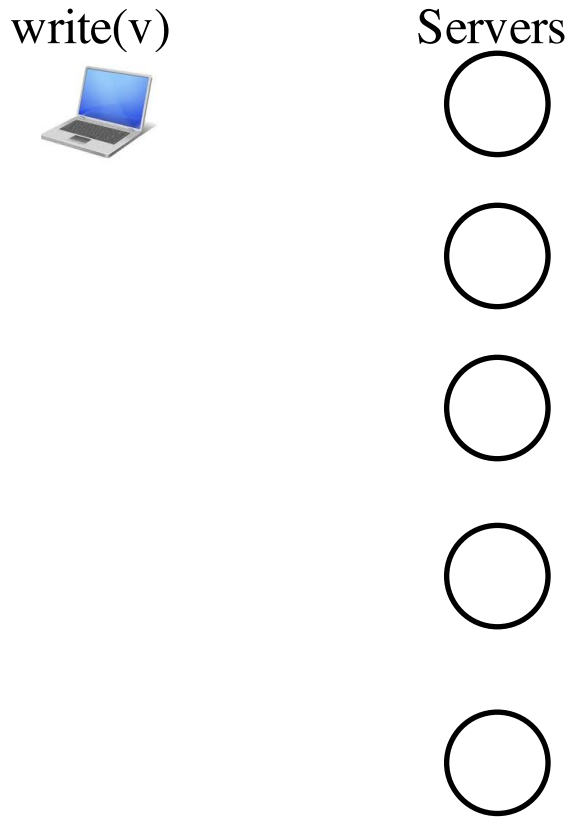
Consistency Model: Atomicity/Linearizability

- Provides the illusion that operations happen in a sequential order
  - a read returns the value of the preceding write
  - a read returns a value at least as recent as that returned by any preceding read



Algorithm ABD: Recalling the past

- [Attiya, Bar-Noy, Dolev 1996] (Dijkstra Prize 2011)
- Order Operations by using <ts, v> pairs.



Writer Protocol

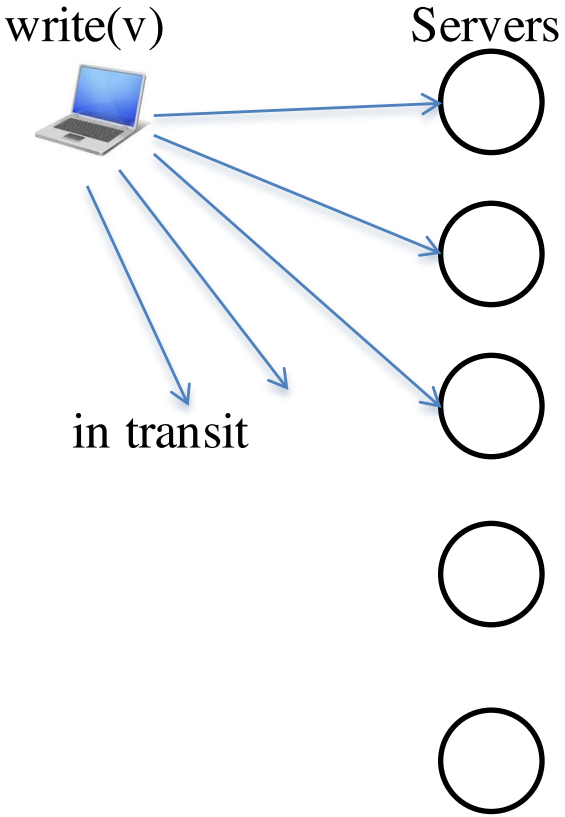
- ts++ //increment ts
- Send <ts,v> to all
- Wait for a majority to reply

Server Protocol (Upon rcv a msg m from i)

- if m.ts > ts
  - Update local <ts,v>
- Send local <ts,v> to i

Algorithm ABD: Recalling the past

- [Attiya, Bar-Noy, Dolev 1996] (Dijkstra Prize 2011)
- Order Operations by using <ts, v> pairs.

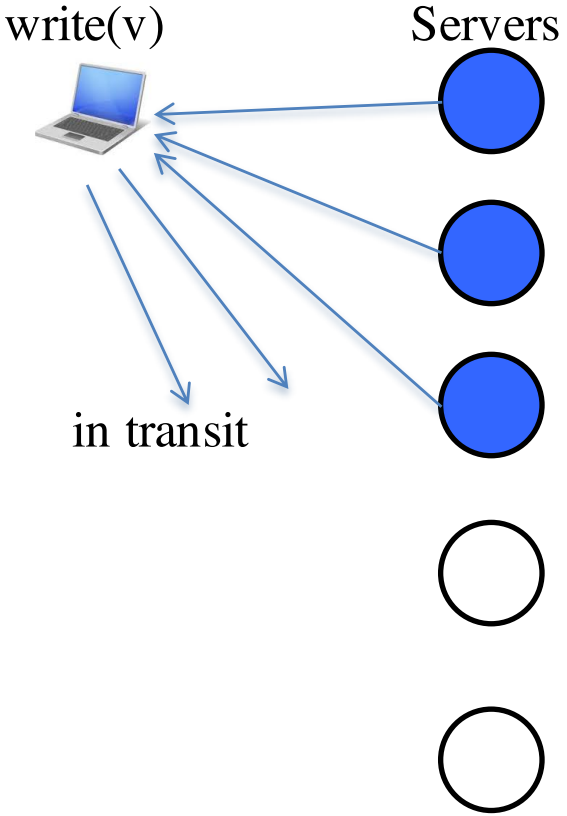


Writer Protocol

- `ts++ //increment ts`
- `Send <ts,v> to all`
- `Wait for a majority to reply`

Algorithm ABD: Recalling the past

- [Attiya, Bar-Noy, Dolev 1996] (Dijkstra Prize 2011)
- Order Operations by using <ts, v> pairs.

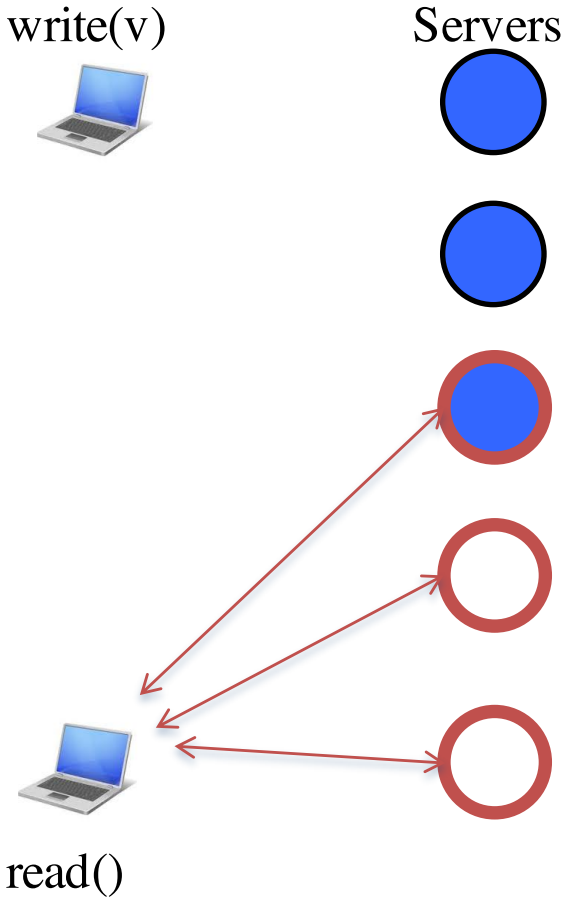


Writer Protocol

- `ts++ //increment ts`
- `Send <ts,v> to all`
- `Wait for a majority to reply`

Algorithm ABD: Recalling the past

- [Attiya, Bar-Noy, Dolev 1996] (Dijkstra Prize 2011)
- Order Operations by using <ts, v> pairs.

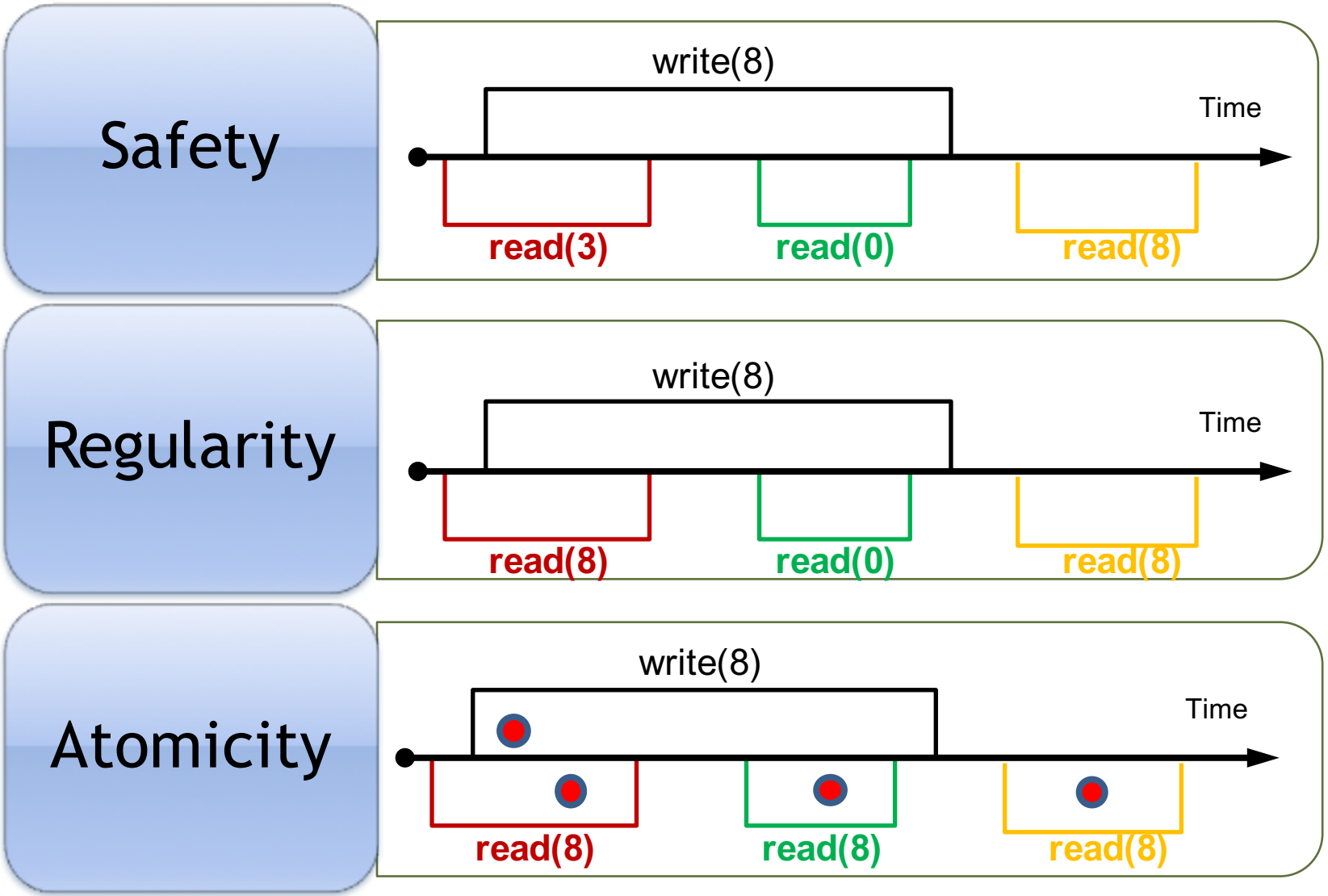


Reader Protocol (2 phases)

- Phase 1:
  - Send read to all
  - Collect <ts,v> from a majority
  - Discover max(<ts,v>)
- Phase 2:
  - Send max(<ts,v>) to all
  - Collect ack from a majority and return v

Reads must Write!

Consistency Semantics [Lamport86]



Definition: Fastness

- 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
- Fast Operation
  - Completes at the end of its first round
- Fast Implementation
  - All operations are fast
- Communication scheme
  - Message delivery: Servers to Clients
  - No server to server or client to client communicaiton