SAPIENZA
UNIVERSITÀ DI ROMA

MIDLAB Middleware Laboratory

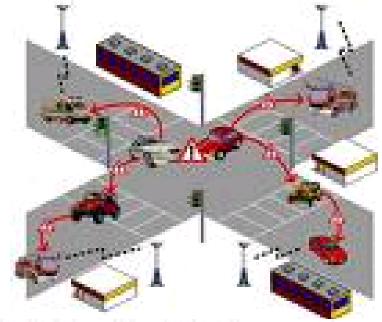
Byzantine Fault Tolerance in Dynamic Distributed Systems

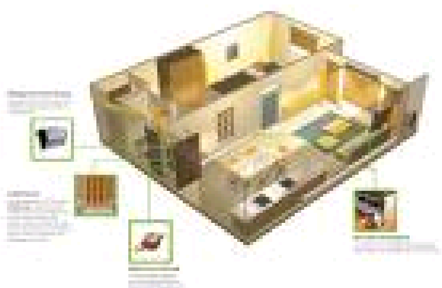
Roberto Baldoni
Università di Roma “La Sapienza”

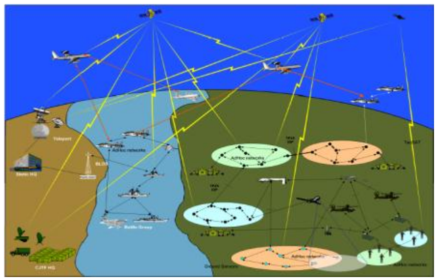
IFIG 10.4 WG 2012 winter meeting
January 27th, La Martinique, France

Advent of Complex Distributed Applications

- Peer-to-peer
- Sensor Networks
- Mobile networks
- Cloud computing federations
- Internet supercomputing
- Smart environments

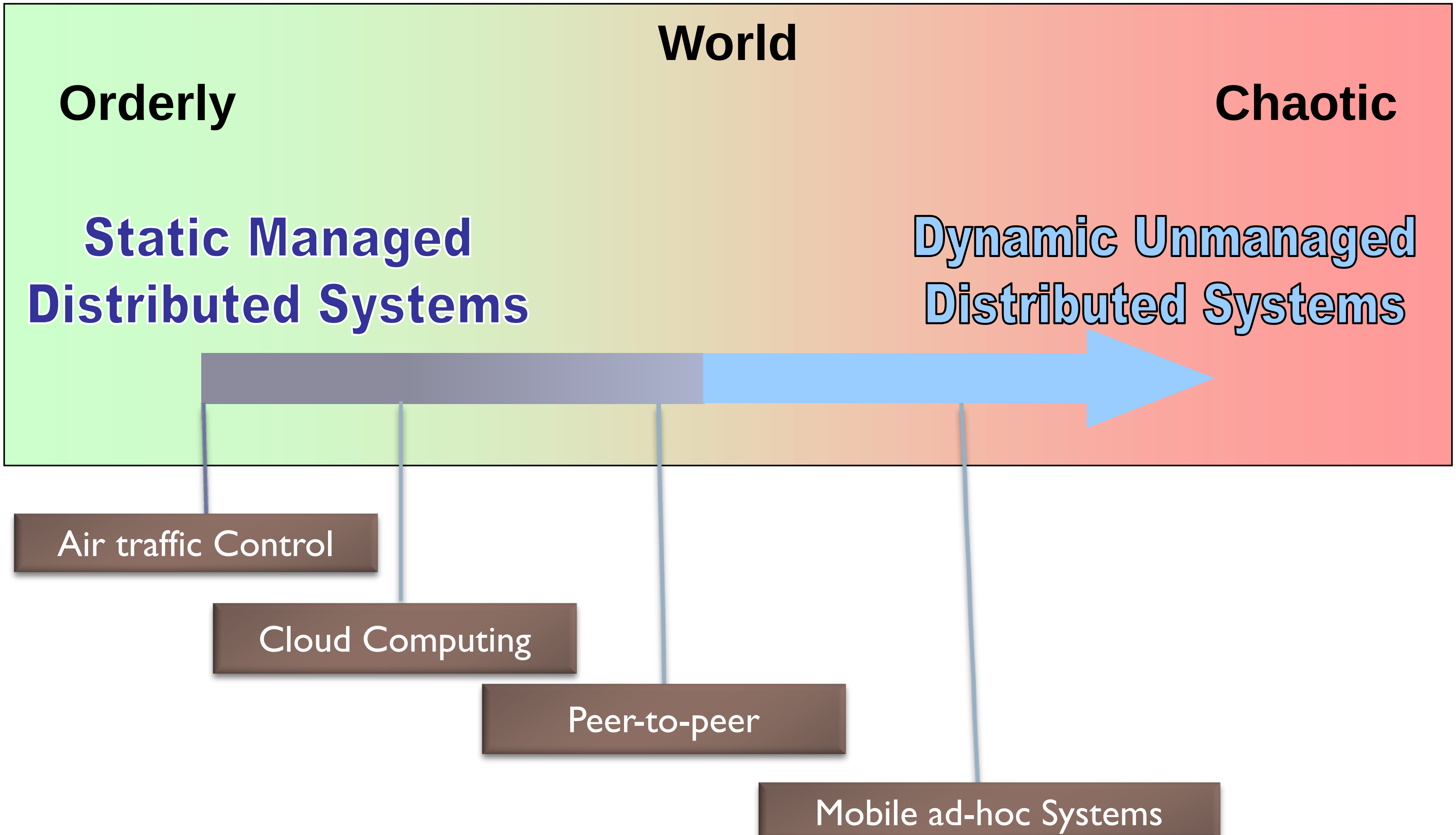






Online Social Networks

Spectrum of Possible System Models



Uncertainty in Dynamic Distributed Systems

- Static Distributed Systems:
  - Lack of temporal knowledge
  - Failures (including byzantine ones)
  - Unknown communication delays
- Dynamic Distributed Systems
  - Same issues as in static distributed systems, plus
  - Non-monotonic and unknown size of the system
  - Potentially changing properties of the "universe"
  - Unclear notions of efficiency, effectiveness, scalability

• Solid theoretical foundations
• Precise problem specifications
• Rigorously correct solutions

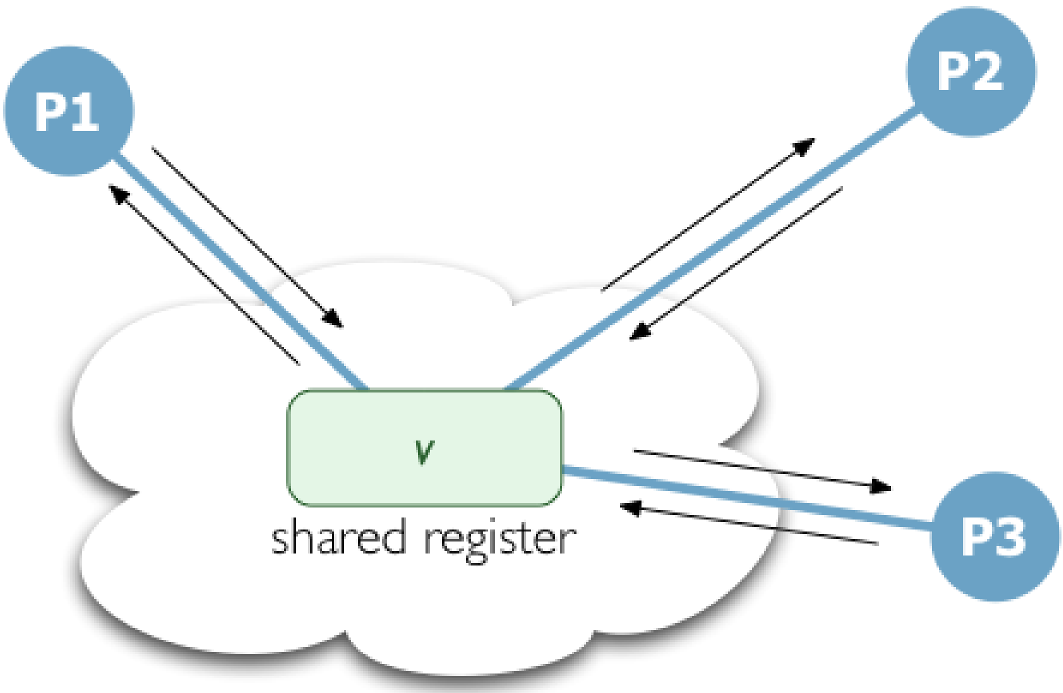
Distributed Storage Service

- Distributed Storage Service is one of the fundamental abstractions to build dependable applications
  - Main requirements: availability, consistency, robustness

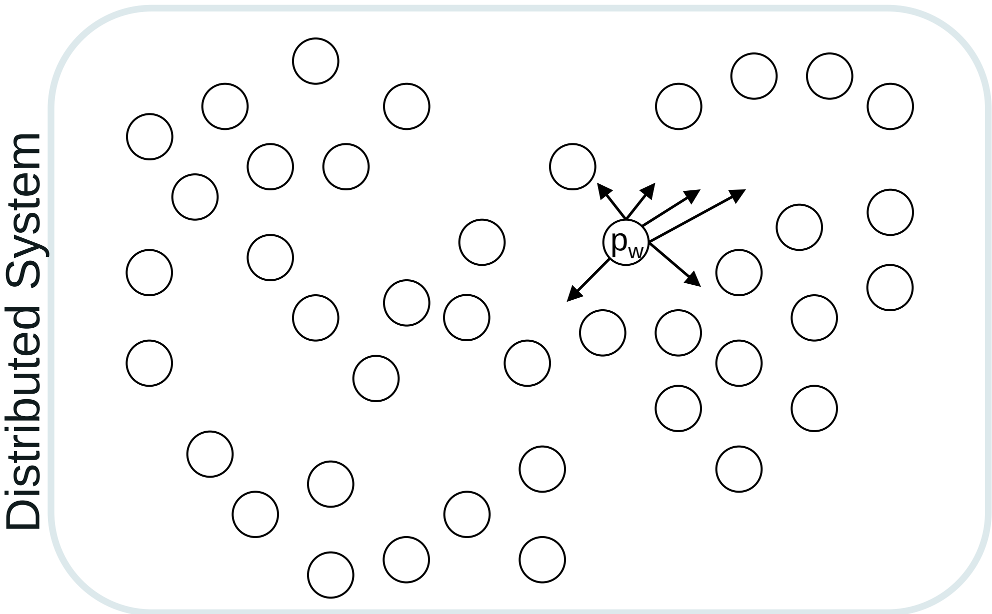
- Modern distributed systems that host storage services are exposed to several vulnerabilities:
  - Asynchrony
  - Crash Failures
  - Attacks from malicious processes (i.e. byzantine failures)
  - maintenance procedures produce churn

Object Abstraction: The Regular Register

A *register* is a shared variable accessed by processes through *read* and *write* operations



Regular Register: write()



The writer process p_w wants to write the value v

A subset of processes participate to the register computation

p_w sends a broadcast message (WRITE, v, sn)

… in the meanwhile processes join and leave the computation

OBS. Only processes belonging to the computation when p_w starts the write and that remain in the computation for all the time of the write will maintain the updated copy of the register

Active Processes keeps the state of the computation

BFT storage in Static Distributed Systems

- State Machine Replication Approach
  - [3] uses 2f + 1 server replicas, and requires that every non-faulty replica agrees to process requests in the same order.
- Quorum Based Approach
  - [1] wait-free single-writer/multi-reader atomic register
    - $n \geq 3f+1$
    - two-phase reading and two-phase writing
  - [2] safe variable with assuming at least 5f replicas
    - $n \geq 5f$
    - one-phase reading and one-phase writing

[1] Aiyer A. S., Alvisi L., Bazzi R. A., “Bounded Wait-Free Implementation of Optimally resilient Byzantine Storage without (Unproven) Cryptographic assumptions”, DISC 2007
[2] Malkhi D., Reiter M.K., “Byzantine Quorum Systems”, Distributed Computing 1998
[3] Schneider Fred B. , “Implementing Fault-Tolerant Services Using the State Machine Approach”, ACM Computing Surveys 1990

Storage Service in Dynamic Distributed Systems

- [1] presents a Reconfigurable Atomic Memory for Basic Object (RAMBO) on top of a distributed systems where processes can join or fail by crashing.
  - Based on Consensus
- [2] shows that a crash resilient atomic register can be realized without consensus, and thus implementable on a fully asynchronous distributed system
  - Assumption of majority of correct processes in any reconfiguration
- [3] provides a crash-resilient regular register in the presence of churn

[1] Lynch, N. and Shvartsman A., "RAMBO: A Reconfigurable Atomic Memory Service for Dynamic Networks", DISC 2002
[2] Aguilera M. K., Keidar I., Malkhi D., Shraer A., "Dynamic atomic storage without consensus", PODC 2009
[3] Baldoni R., Bonomi S., Kermarrec A.M., Raynal M., "Implementing a Register in a Dynamic Distributed System", ICDCS 2009

Related work

1. BFT Registers in Distributed Systems with churn
    1. Single-writer/multi-reader Regular Register [1]
        1. Assumption of bounded execution time for each operation
        2. Bound on the churn depending on the duration of each operation
    2. multi-writer/multi-reader safe register [2]
        1. Synchronous System prone to continuous churn

[1] *Baldoni R., Bonomi S., SoltaniNezhad A., "*Brief Announcement: Validity Bound of Regular Registers with Churn and Byzantine Processes*",* PODC 2011
[2] Bonomi S., Soltani Nezhad A., "*Multi-writer Regular Registers in Dynamic Distributed Systems with Byzantine Failures* ", TADDS2011

System Model

Clients System

- ✓ Composed by a finite arbitrary number of processes

Servers System

- ✓ It is dynamic
  - ✓ New servers are connected along time
  - ✓ Servers can be disconnected

Autenthicated Communication

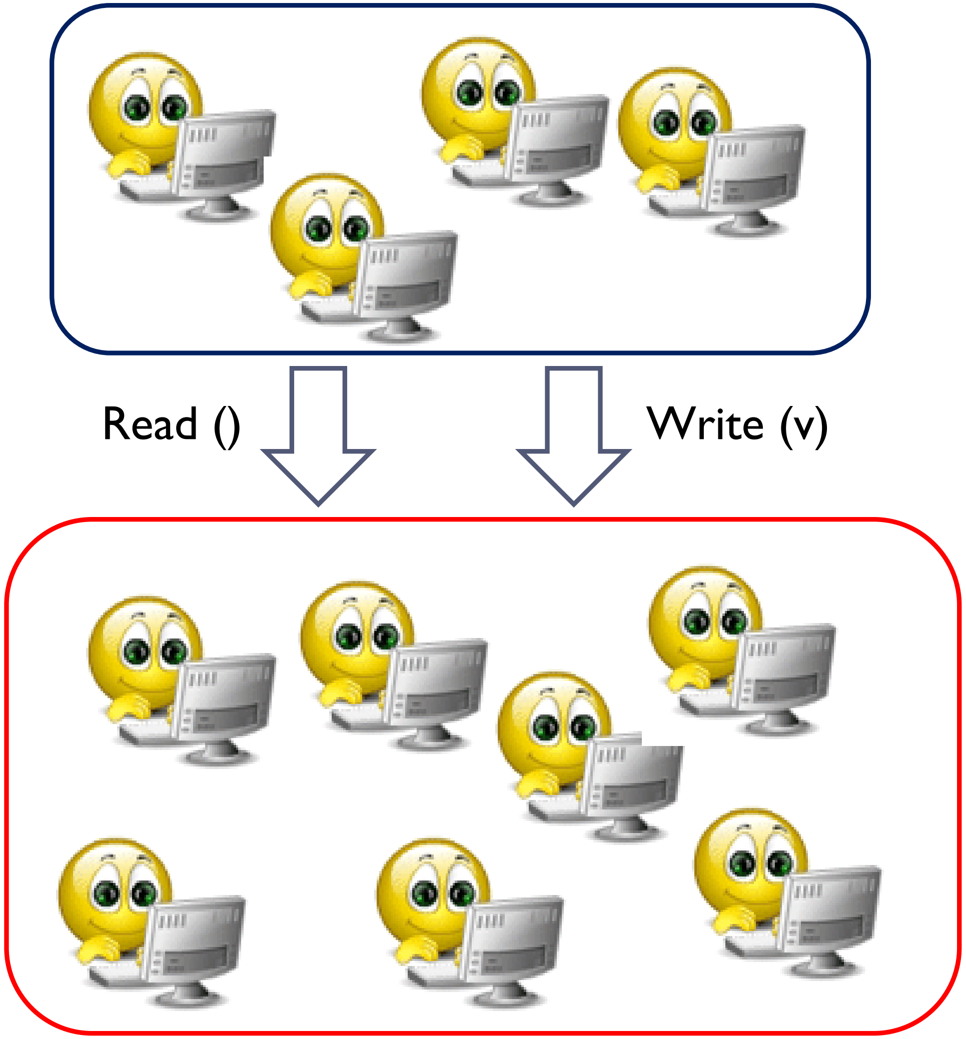
Impossibility of realizing a regular register

- [1] shows that it is not possible to implement a register in a fully asynchronous distributed system prone to continuous churn

- Eventual Synchrony Assumption
  - There exist a time t such that any message m broadcast/sent from a process pi, at some time t'>t is delivered by time t'+δ unless pi leaves the computation between t' and t'+δ.

[1] Baldoni R., Bonomi S., Raynal M., ": Implementing a Regular Register in an Eventually Synchronous Distributed System Prone to Continuous Churn", IEEE TPDS 2012

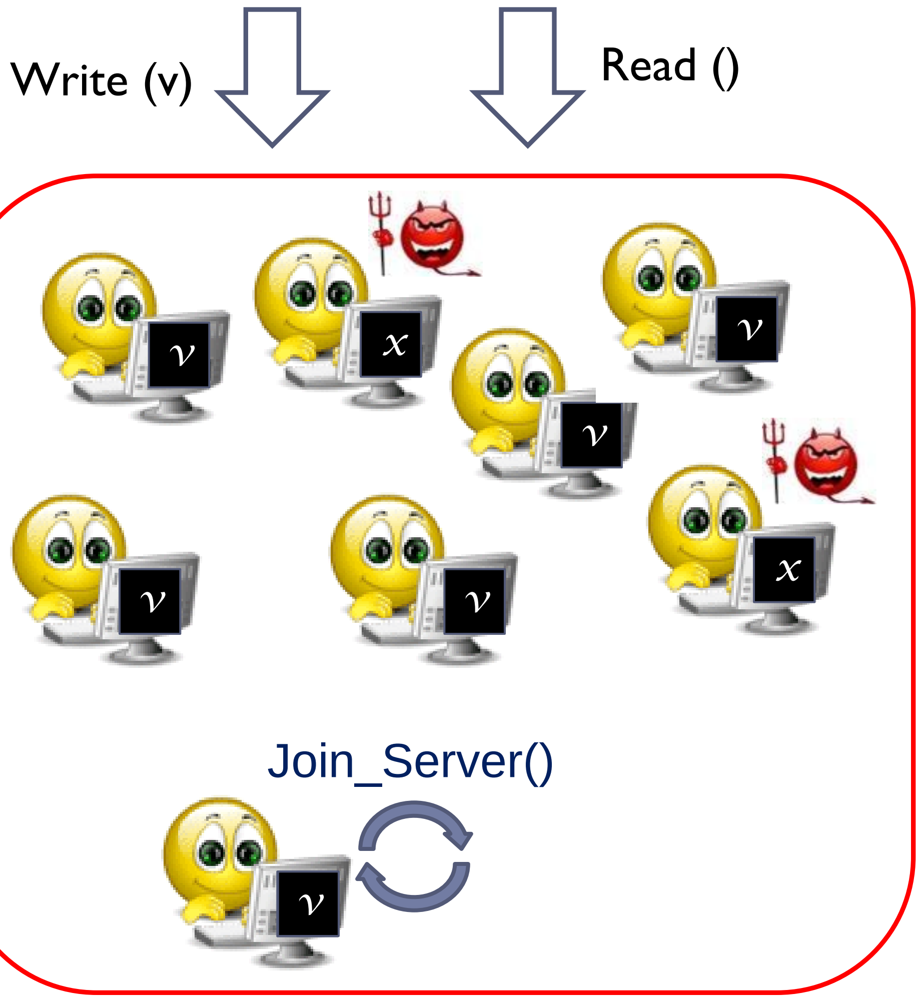
Computation Model



Clients Computation

- ✓ Clients are not byzantine, but can crash
- ✓ No information about register state
- ✓ Clients trigger read() and write() operations

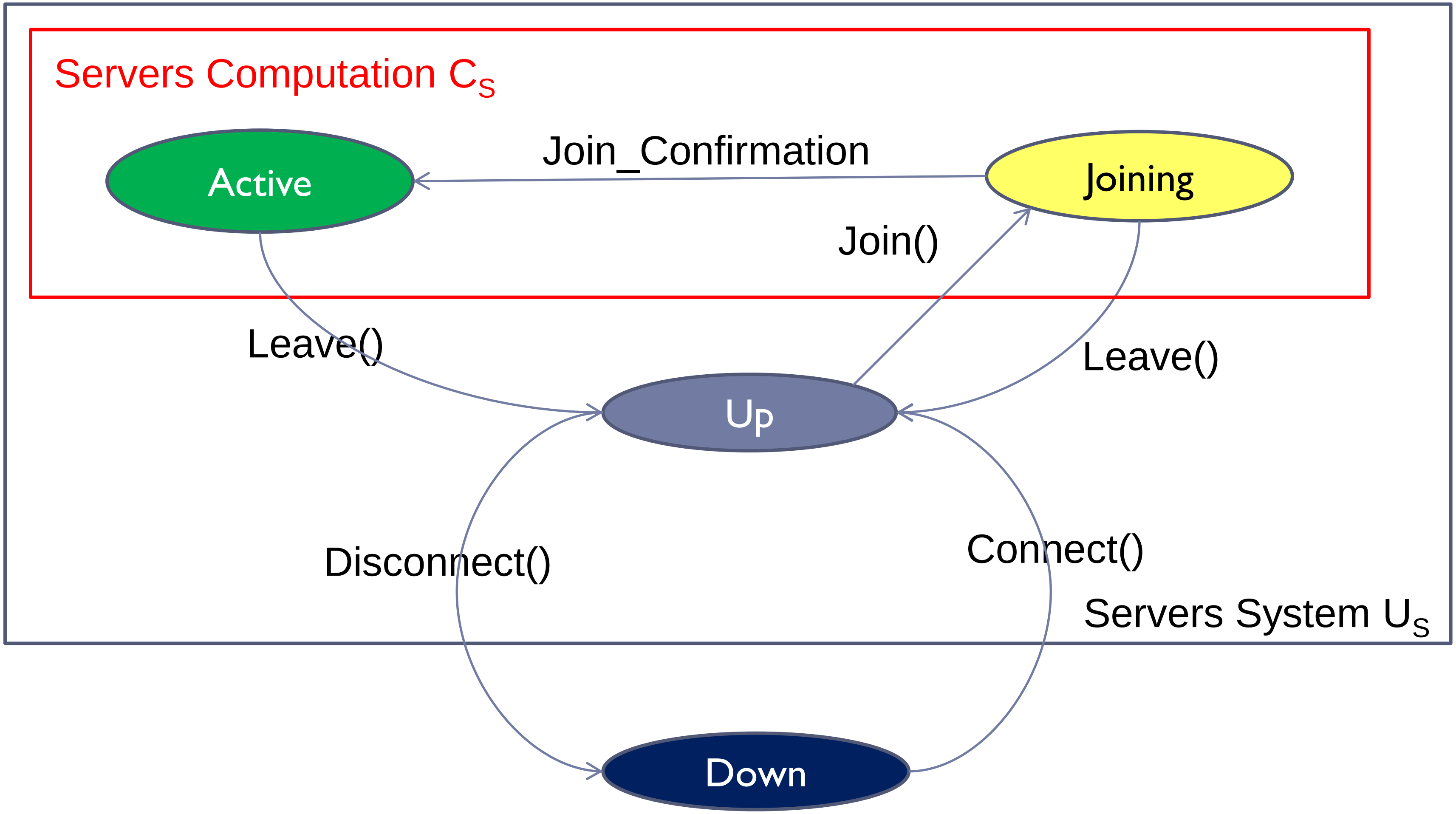
Computation Model



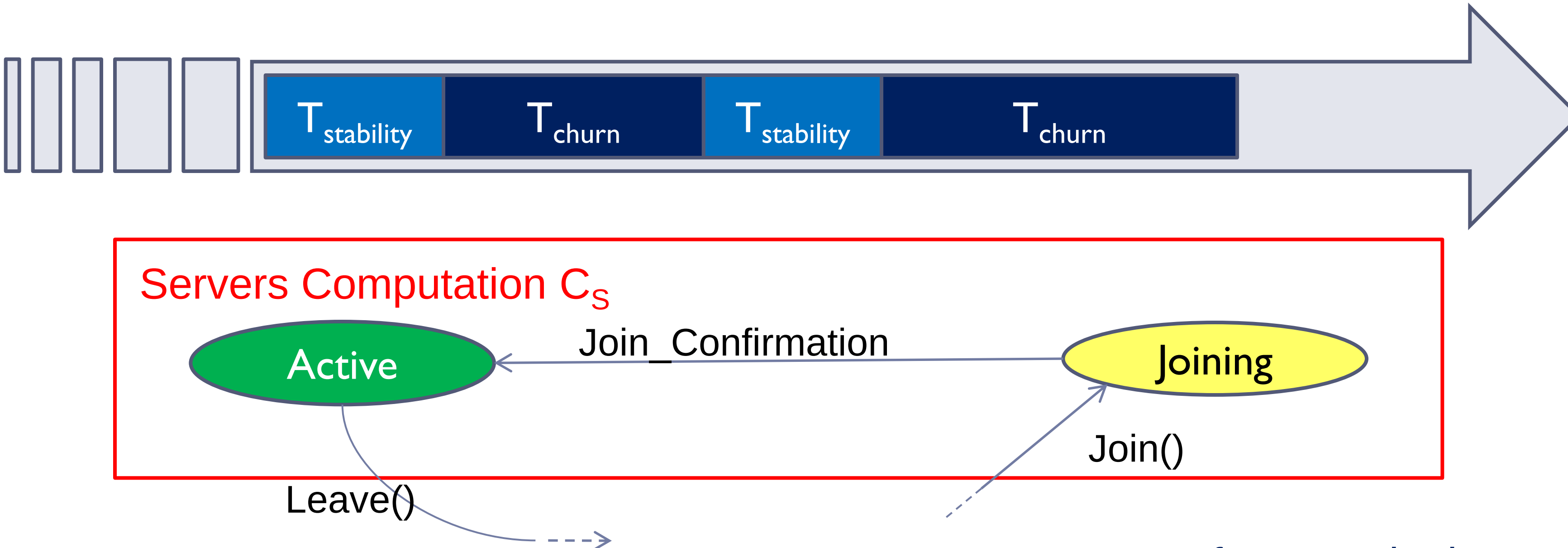
Servers Computation

- ✓ Initially n servers are part of the register computation
- ✓ Servers do not know how is currently in the computation
- ✓ Up to f byzantine failures
- ✓ Servers maintain locally a copy of the register value
- ✓ Alternating periods of churn and stability
  - ✓ No stable processes
  - ✓ In churn periods the servers set is continuously changing

Correct Servers' Life Cycle



Churn Model



Assumption

$$|A(t)| \geq n - J$$

n = # of servers in the computation at time t_0

J = maximum number of concurrent joining processes

$$|C(t)| \in [n, \; n - J]$$

Safe Register Specification

- Termination
  - If a correct process (either a client or a server) participating in the computation invokes an operation and does not leave the system, it eventually returns from that operation
- Validity
  - a read() operation, not concurrent with any write(), returns the last written value before its invocation. In the case of concurrency, a read() may return any value.

Issues

- Byzantine servers
  - Possible collusions to compromise the register state

    ⇓

    - Given f faulty servers, at least 2f+1 values are needed to filter out faulty ones
- Churn
  - The set of replicas maintaining the register value continuously change during time

    ⇓

    - The current value may disappear after a certain amount of time
- Eventually Synchronous Communications

  ⇓

    - write messages can be missed by new servers
    - ack messages can be lost due to servers departures

Algorithm

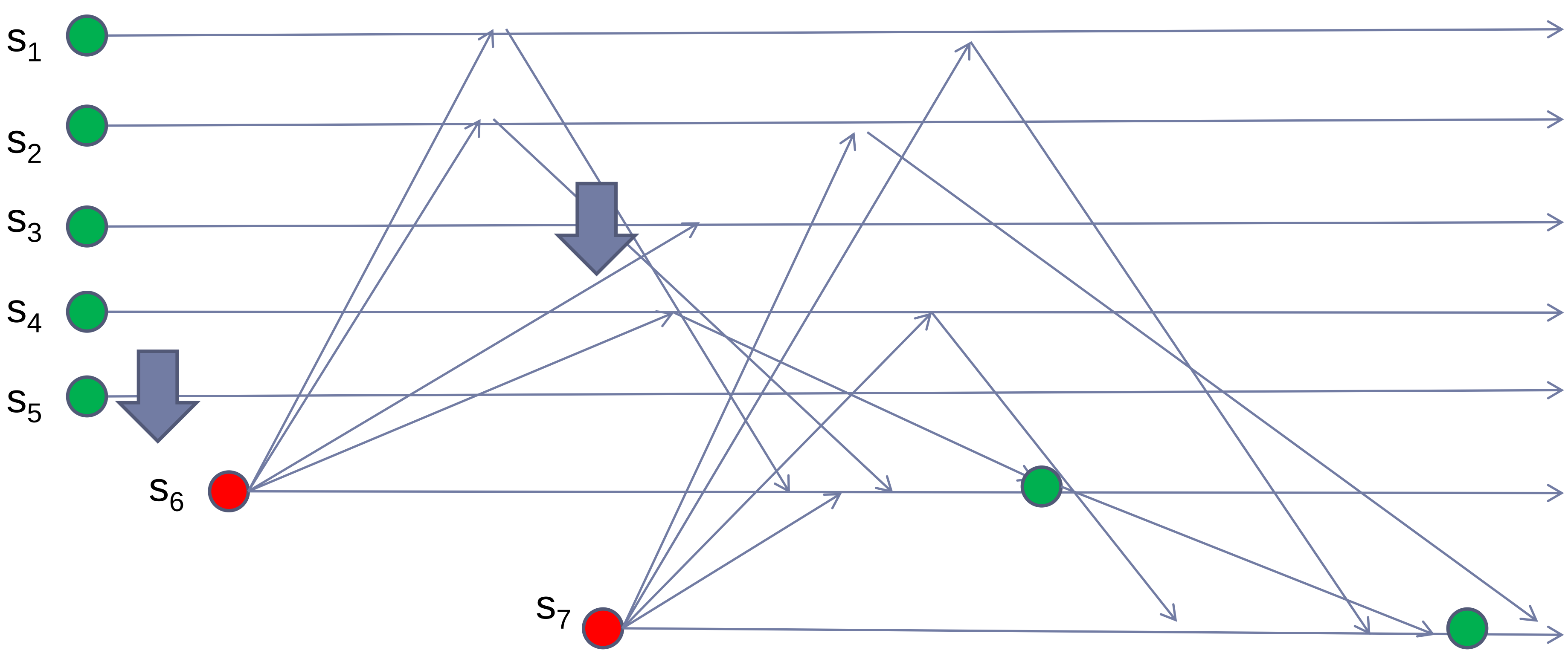
- General Idea:
  - Read and Join Operations should be as fast as possible (1 phase)
  - Extension of Malkhi-Reiter byzantine quorums [1] to distributed system prone to churn
    - Read() and write() operations are performed on a quorum of n-f-J servers
    - The join() operation is a particular case of read

[1] Malkhi D., Reiter M.K., "*Byzantine Quorum Systems*", Distributed Computing 1998

read() and write(v) operations

- write(v)
  - Client periodically sends the value and its timestamp
  - Servers acknowledge the new value
  - Client waits for n-f-J ack and then ask for confirmation
  - Servers confirm the new value
- read()
  - Client periodically ask for the current value
  - Active servers reply with a pair <v, ts>
  - Client waits for n-f-J replies and then select the most frequent value

Join() Operation



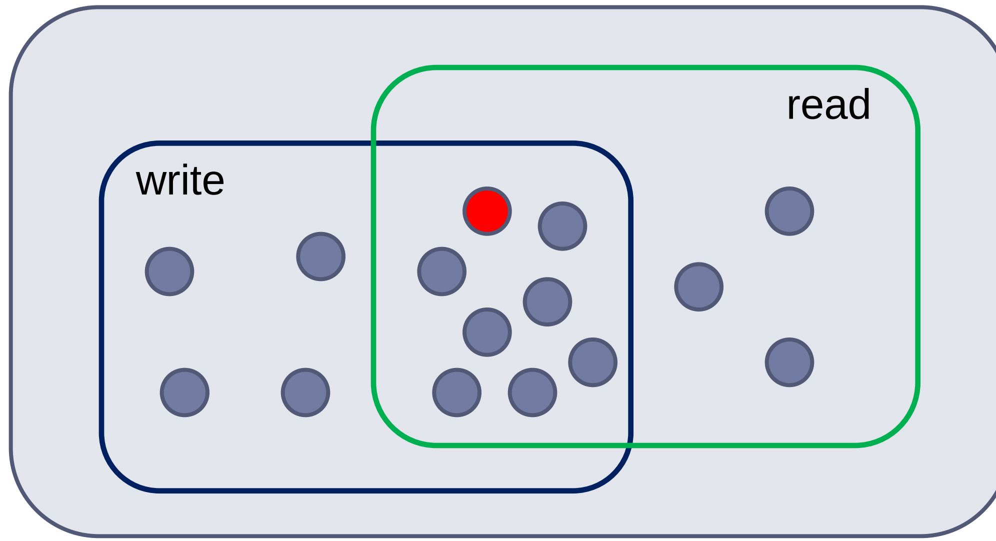
Theorem

- Validity
  - If $n \geq 5f + 3J$, then a read() operation that is not concurrent with any write(), returns the last value written before the read() invocation.

- Termination.
  - Let $n \geq 5f + 3J$. If a process invokes join(), read() or write (), and does not leave the system, it eventually terminates its operation

Correctness (intuition)

- Quorum size is n-f-J
- An Opaque Masking quorum system (as defined in [1]) exists if correct affected servers are more than
  - faulty answering to a read +
  - not affected ones



- Considering that joining processes are not faulty but only temporarily silent, this condition is true for $n \geq 5f + 3J$

- Validity is guaranteed by the existence of the quorum system
- Termination follows from the eventual synchrony

[1] Malkhi D., Reiter M.K., "*Byzantine Quorum Systems*", Distributed Computing 1998

Conclusion and future work

- A safe register can be implemented also in the presence of both bounded churn and byzantine servers
- However…
  - A lot of replicas are needed to cope with this issues

- Some open questions for our future work
  - Can we remove the assumption of bounded f ?
  - Can we handle rational behavior ?
  - Other attacks ?

- Main achievement: churn is a specific type of behavior that have to be handled appropriately!

Thank You!

Questions?!