XCO.T473 サイバーセキュリティ概論

https://www.coord.c.titech.ac.jp/c/cybersec

Intrusion-Tolerant Computer Systems

耐侵入コンピュータシステム

Xavier Défago

School of Computing

Tokyo Institute of Technology

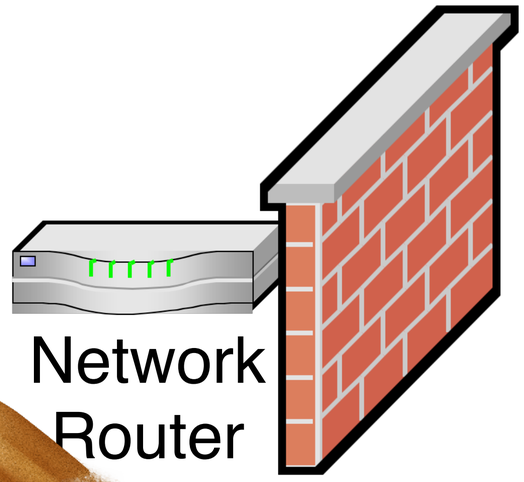
第8回目

Why Intrusion-Tolerance?

Firewall Fallacy

“Inside”

comfy, safe

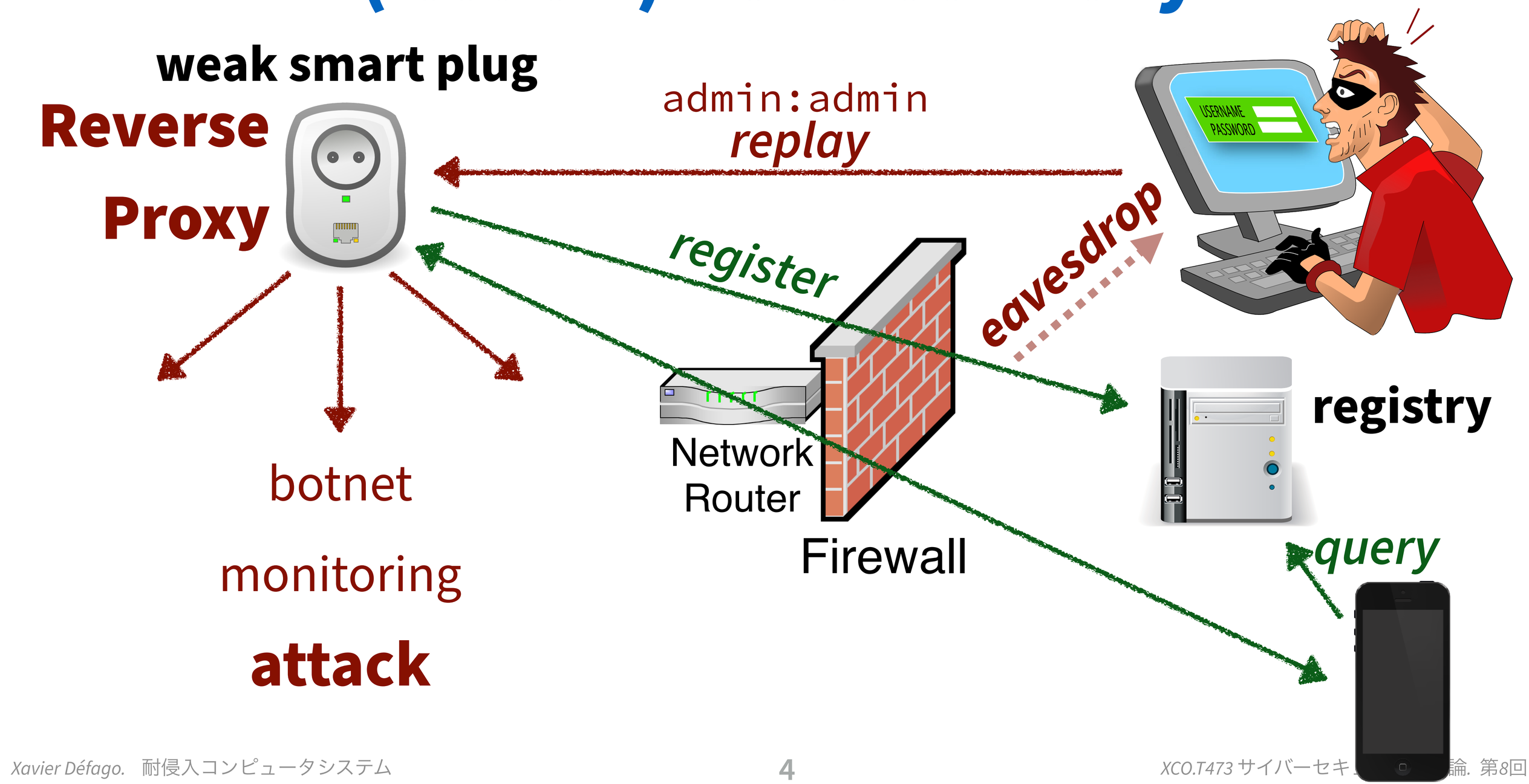


Firewall

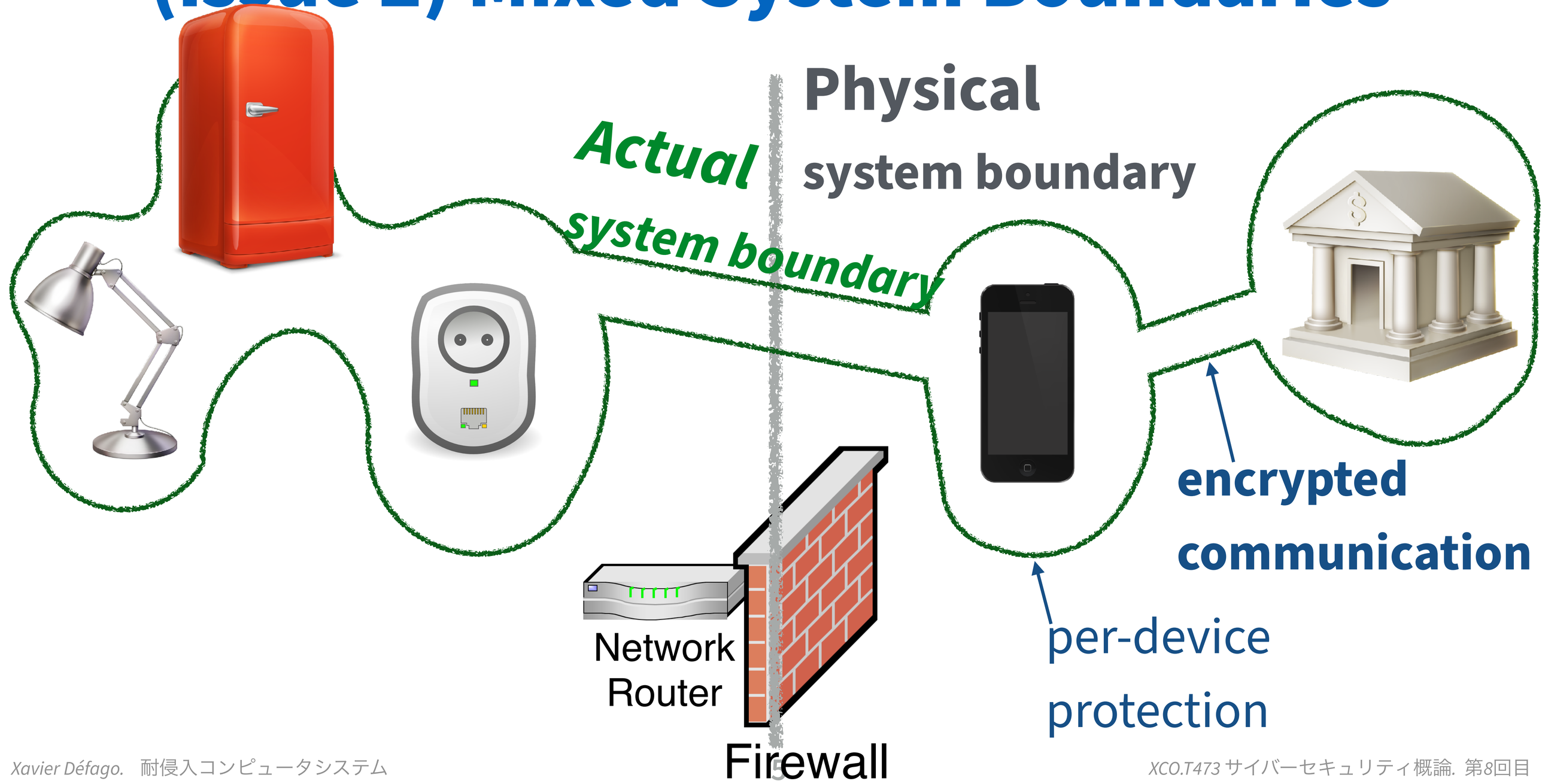
“Outside”

threatening

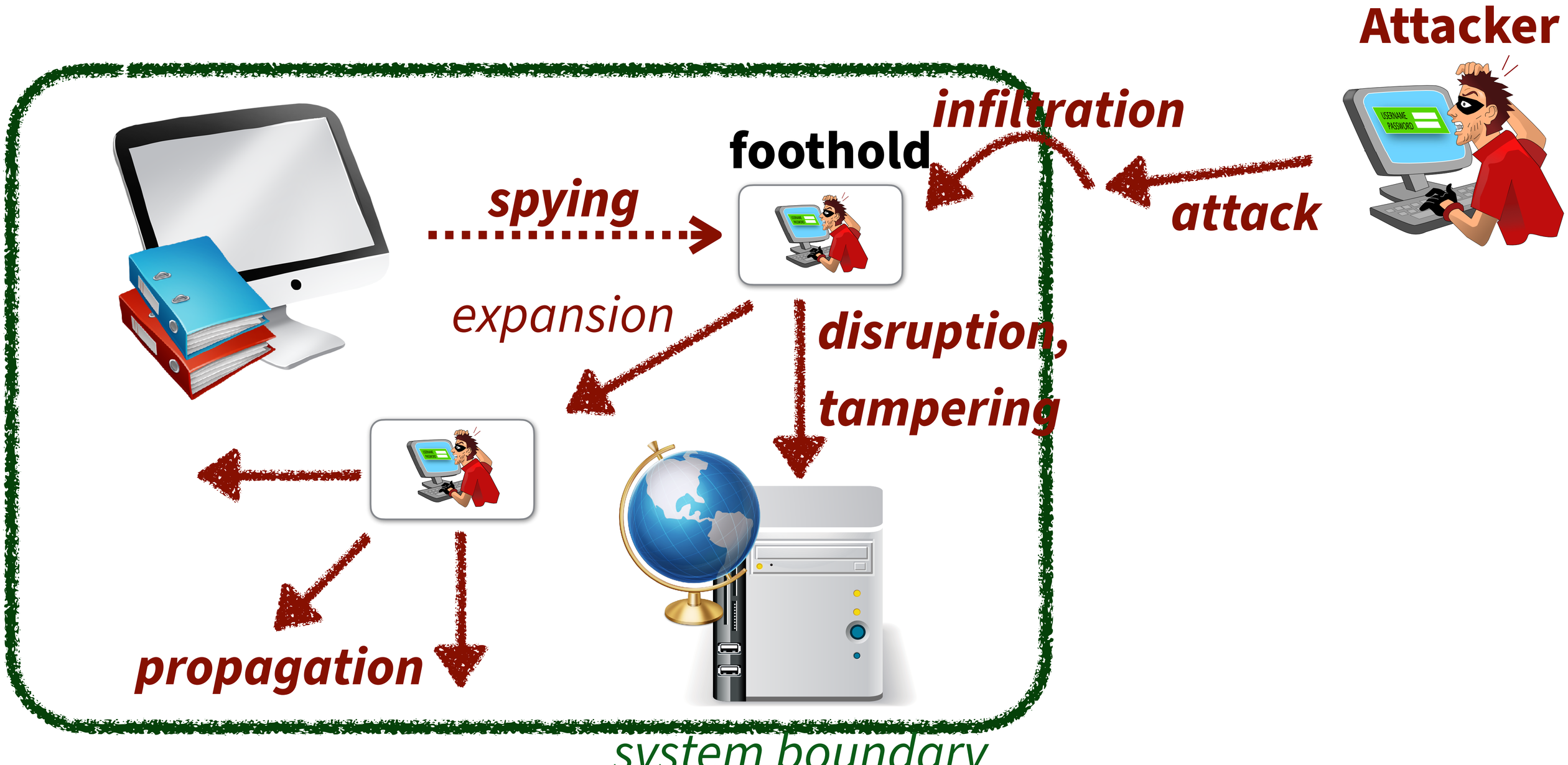
(issue 1) Reverse Proxy



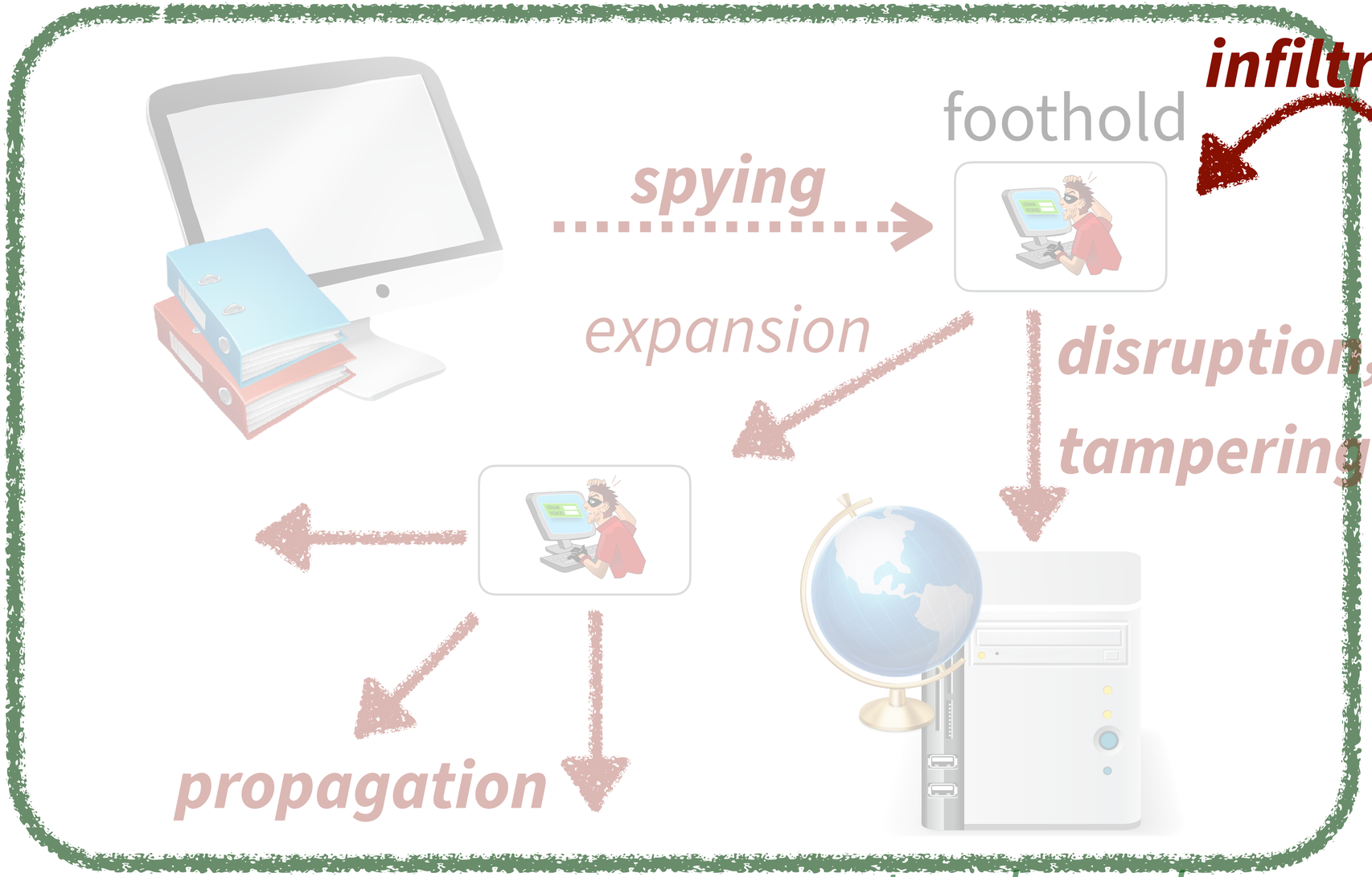
(issue 2) Mixed System Boundaries



Phases of an Attack

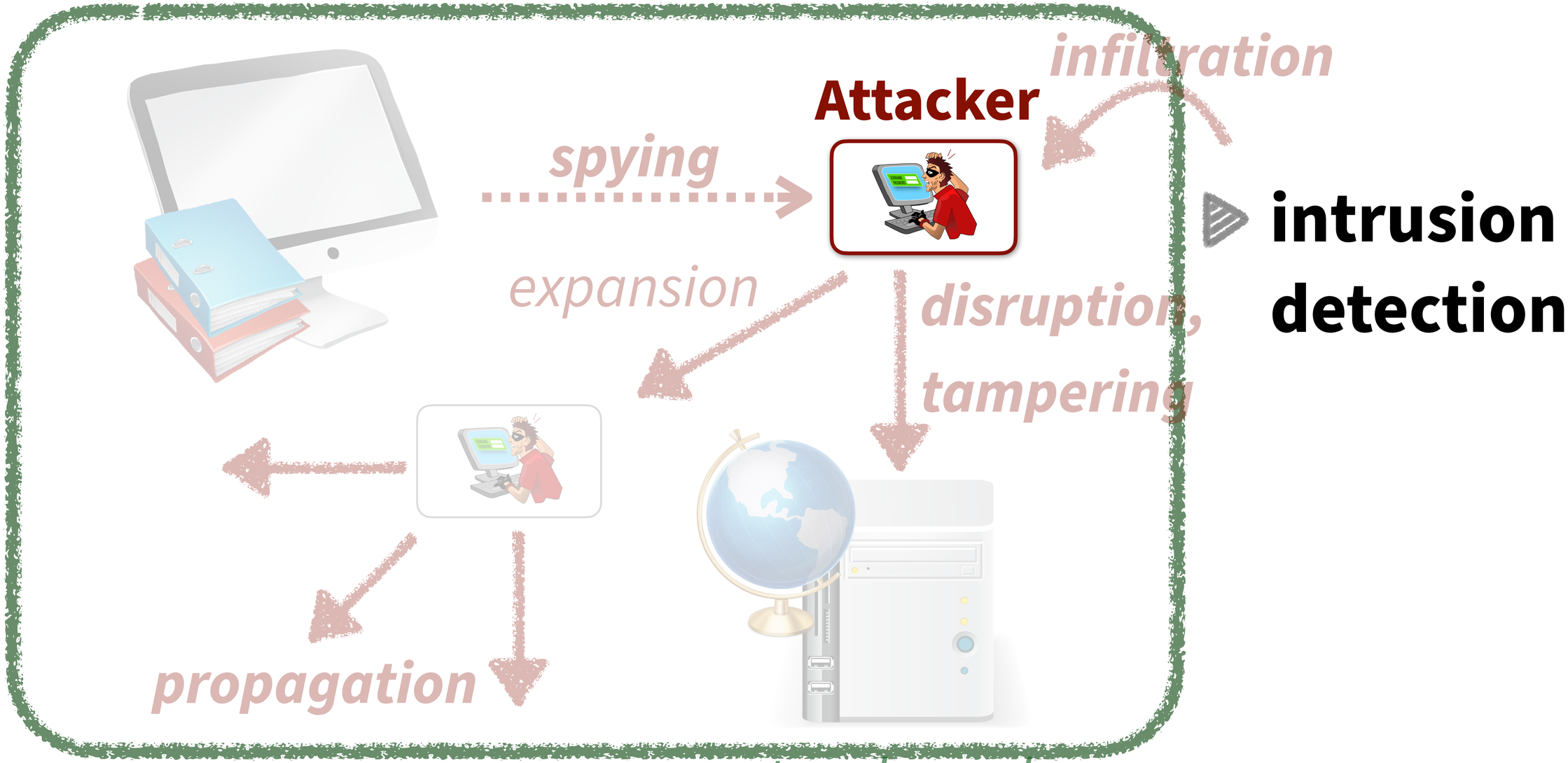


Typical Countermeasures



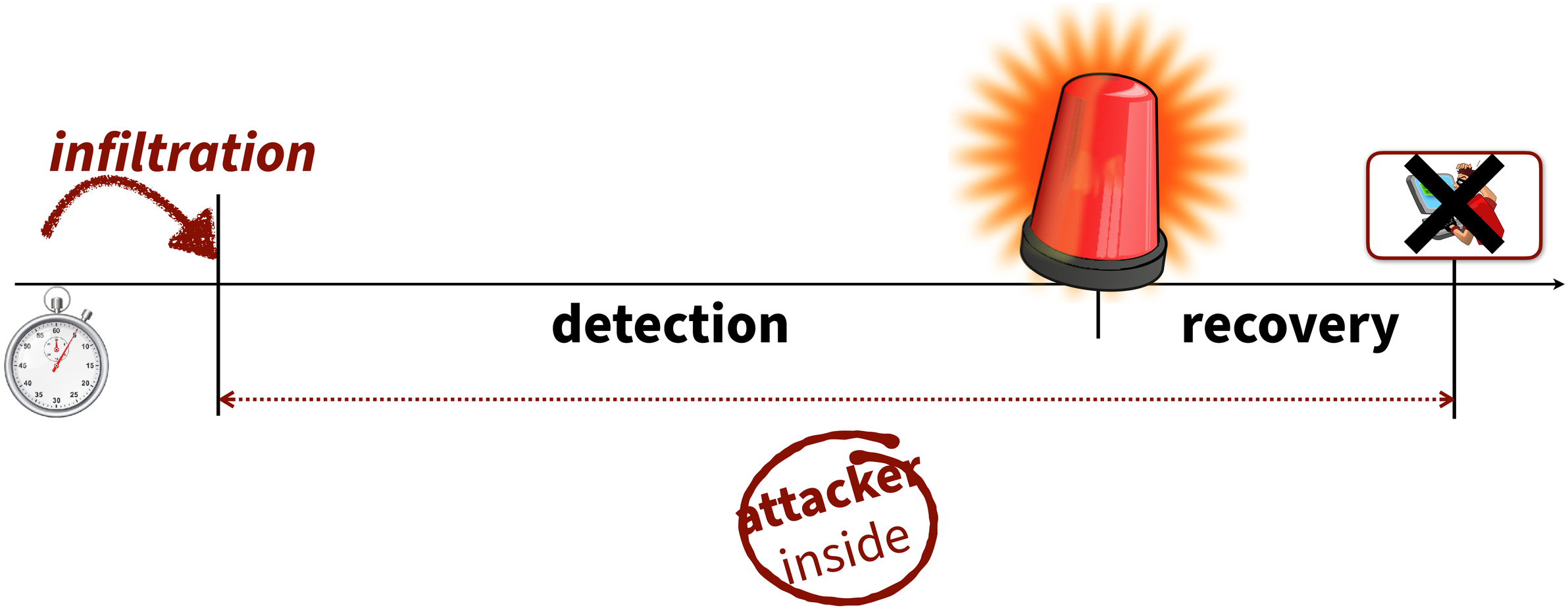
- firewall(s)
- encryption (net.)
- authentication

Typical Countermeasures

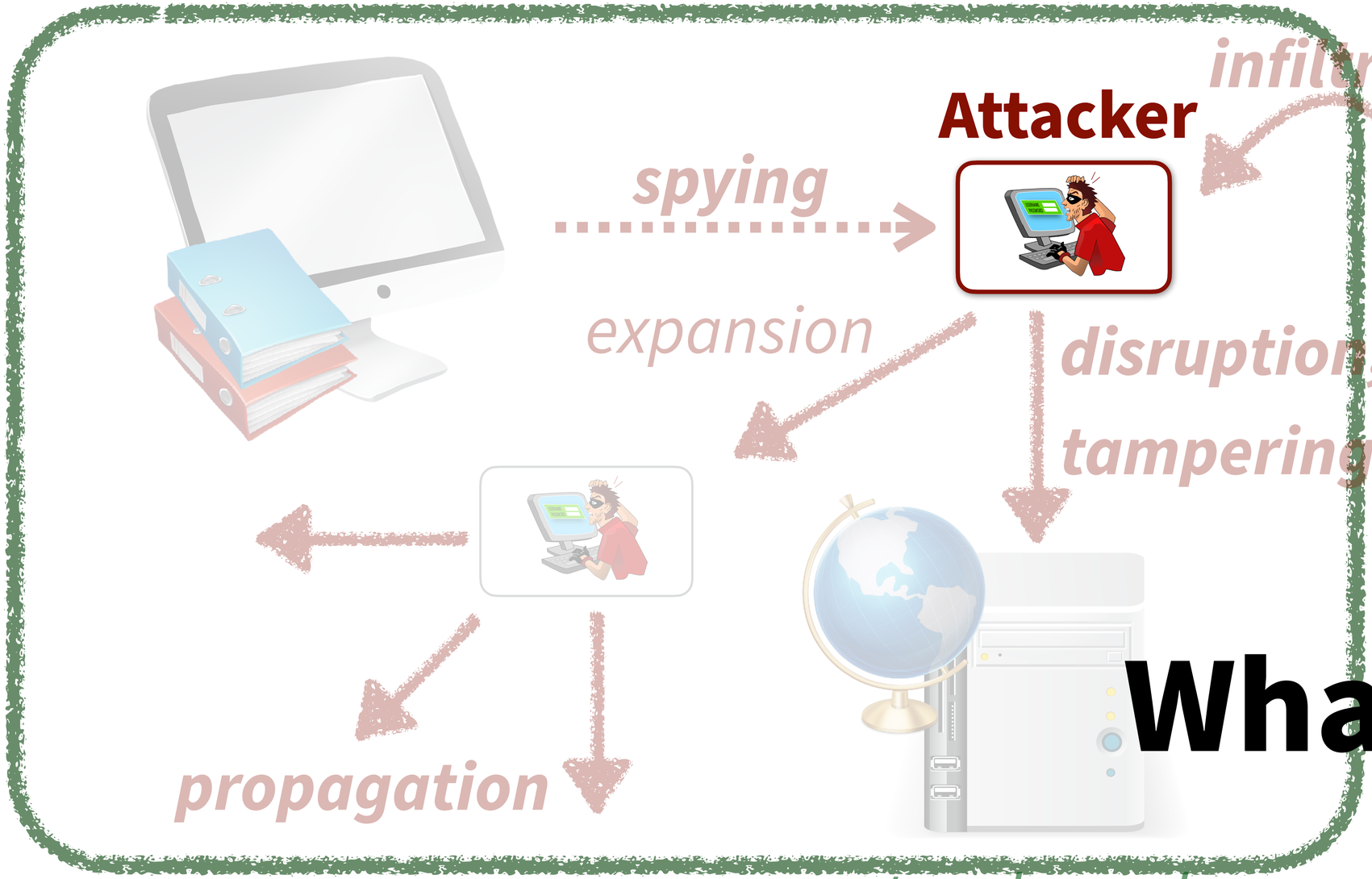


Intrusion Detection

Detection time is very long!



Typical Countermeasures



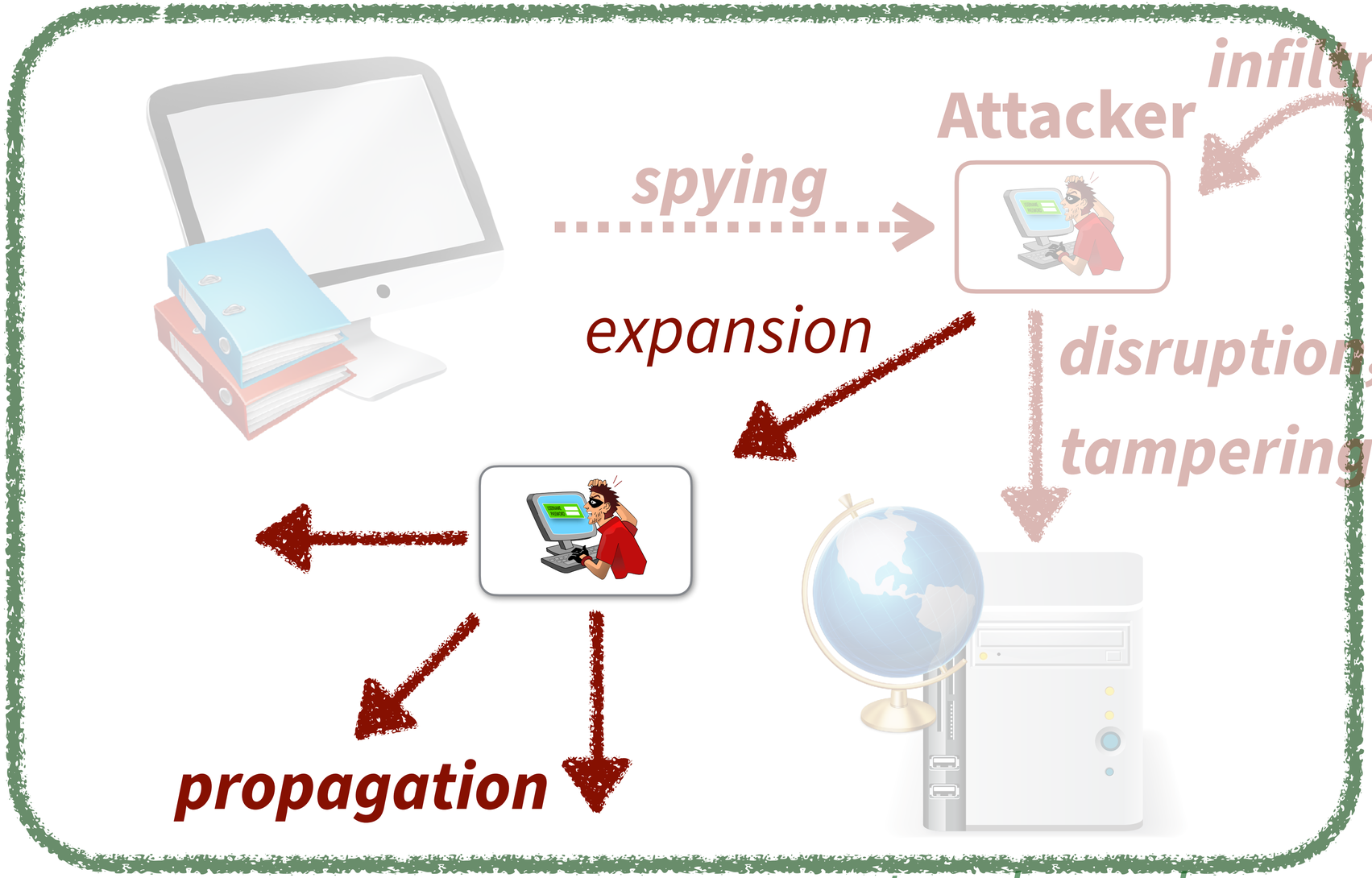
- intrusion detection
- long delays
- meanwhile:

What happens?

Typical Countermeasures

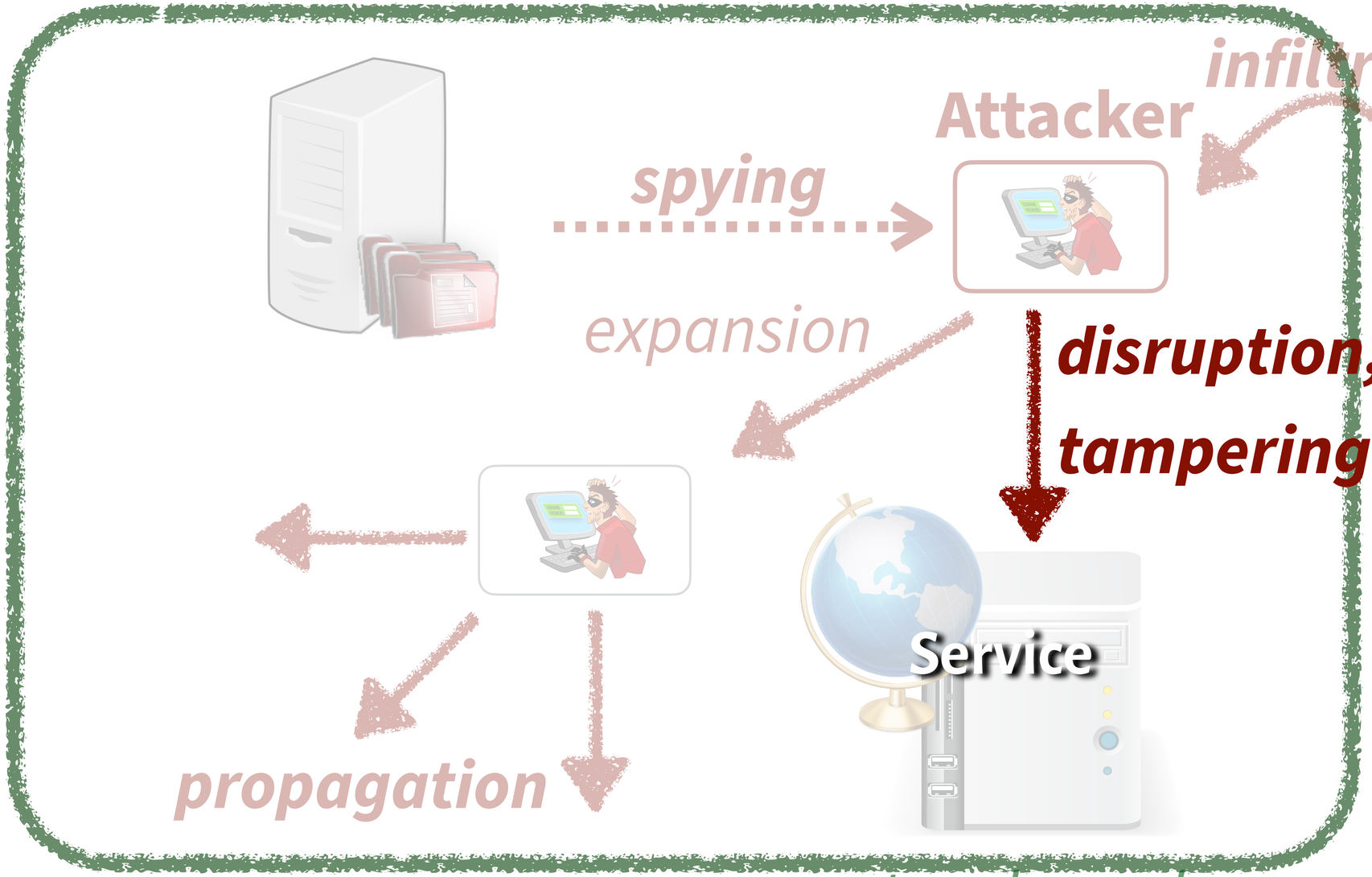
- encryption (data)
- encryption (net.)

Typical Countermeasures



- multi-versioning
- port blocking
- repair / recover

Typical Countermeasures



… none yet …

Typical Countermeasures

Service = ?

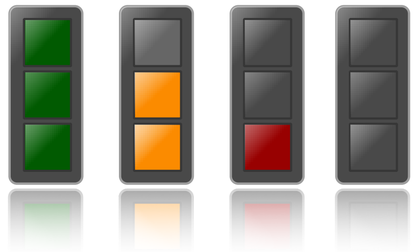
- authentication?
- intrusion detection?
- …

Keep
System Integrity

Aspects of Security

- **Confidentiality**
  - No information leak

- **Availability**
  - Provide operation (no DoS)



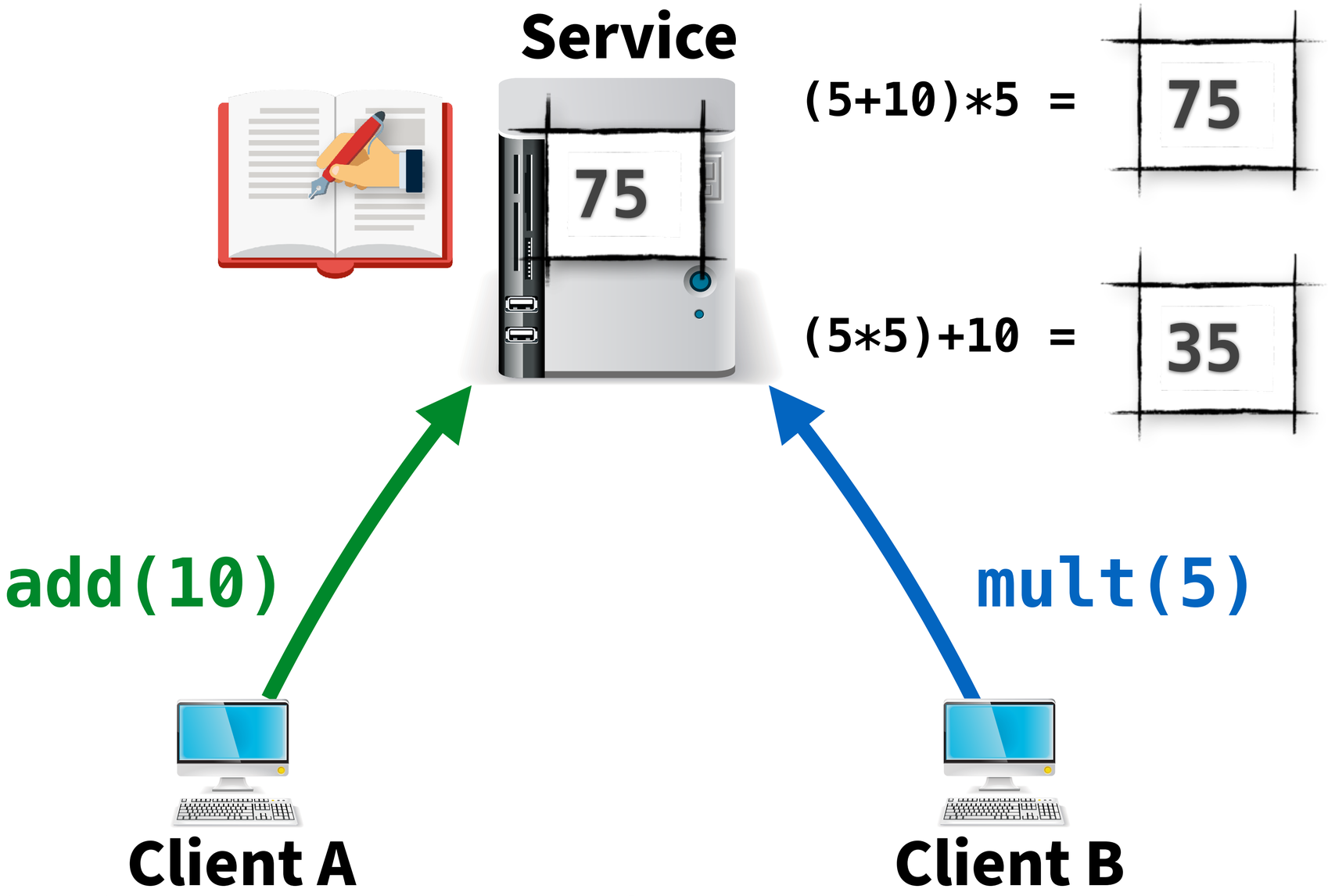
- **Integrity**
  - Prese rrect state

- **Auth on**
  - “Are you?”

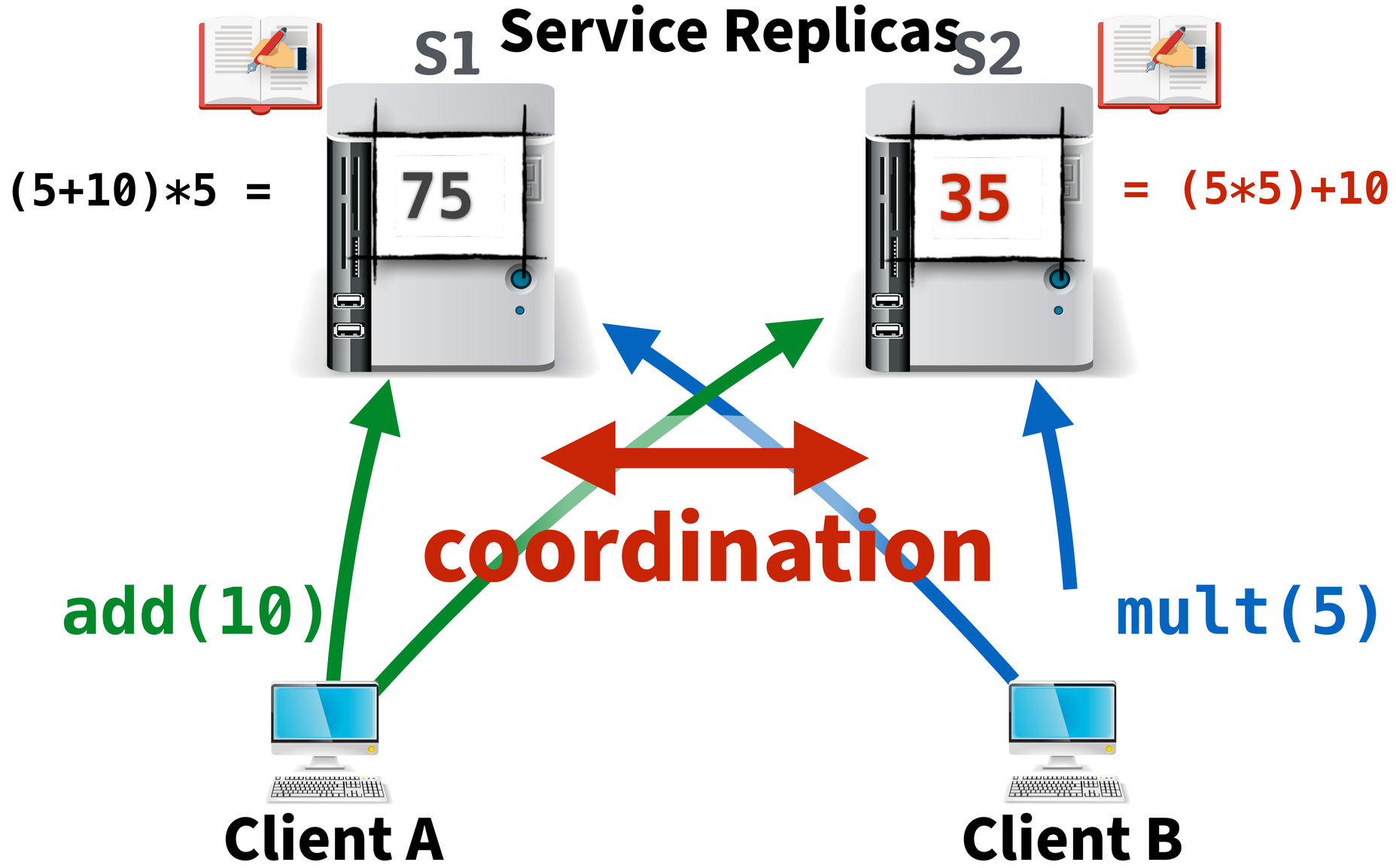
Data / System Integrity

- **Requirements**
  - No tampering of data / system
  - Distributed system: preserve **consistency**
- **“Outside” Threat**
  - Attacker has **no secret** initially
  - Attacker uses **public** service interface in unexpected ways
- **“Inside” Threat**
  - Attacker **passes as legitimate** node / server (eludes detection)
  - Attacker has access to **all information**

The Replicated Register



The Replicated Register

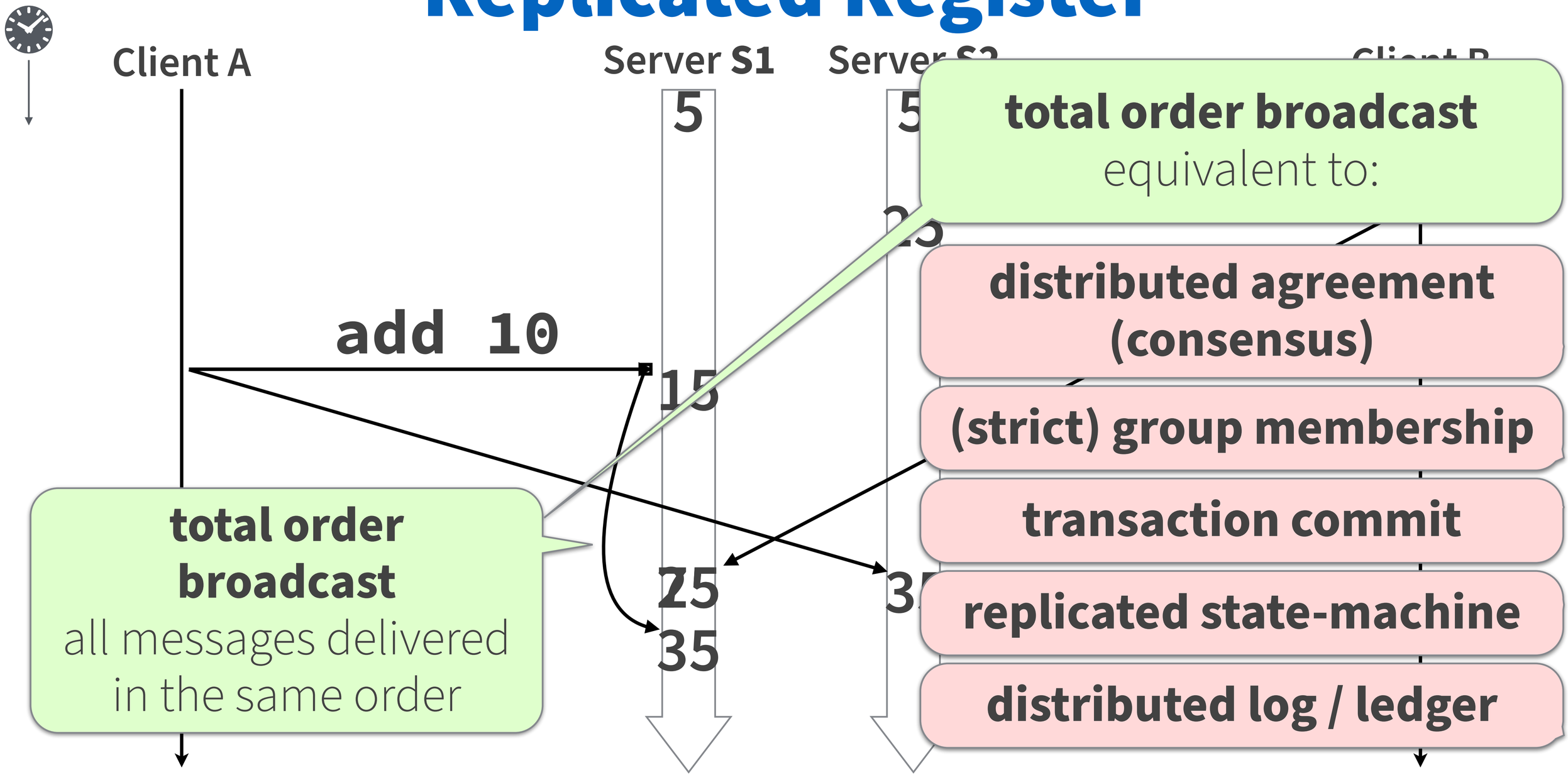


Consistency

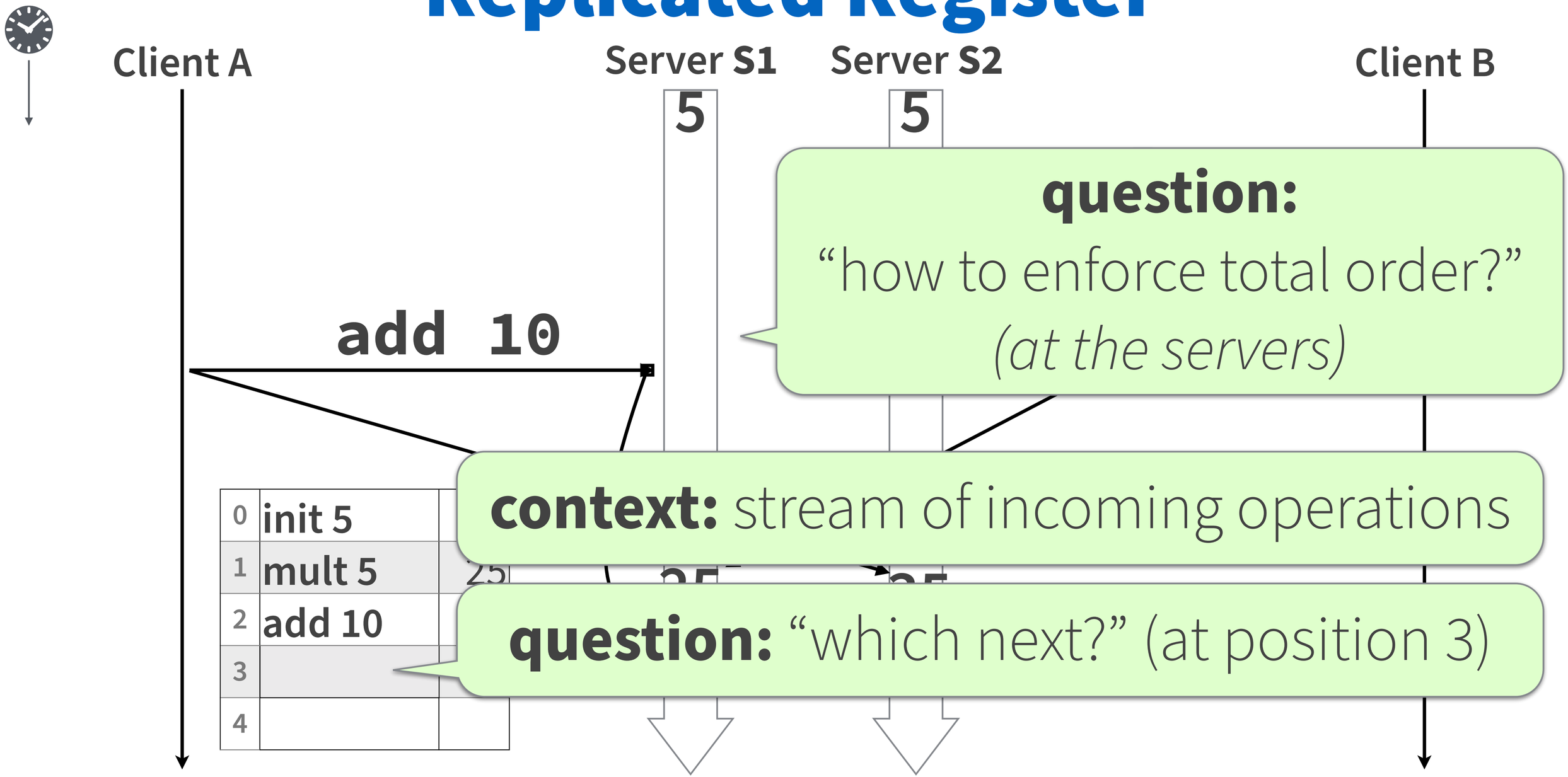
&

Total Order

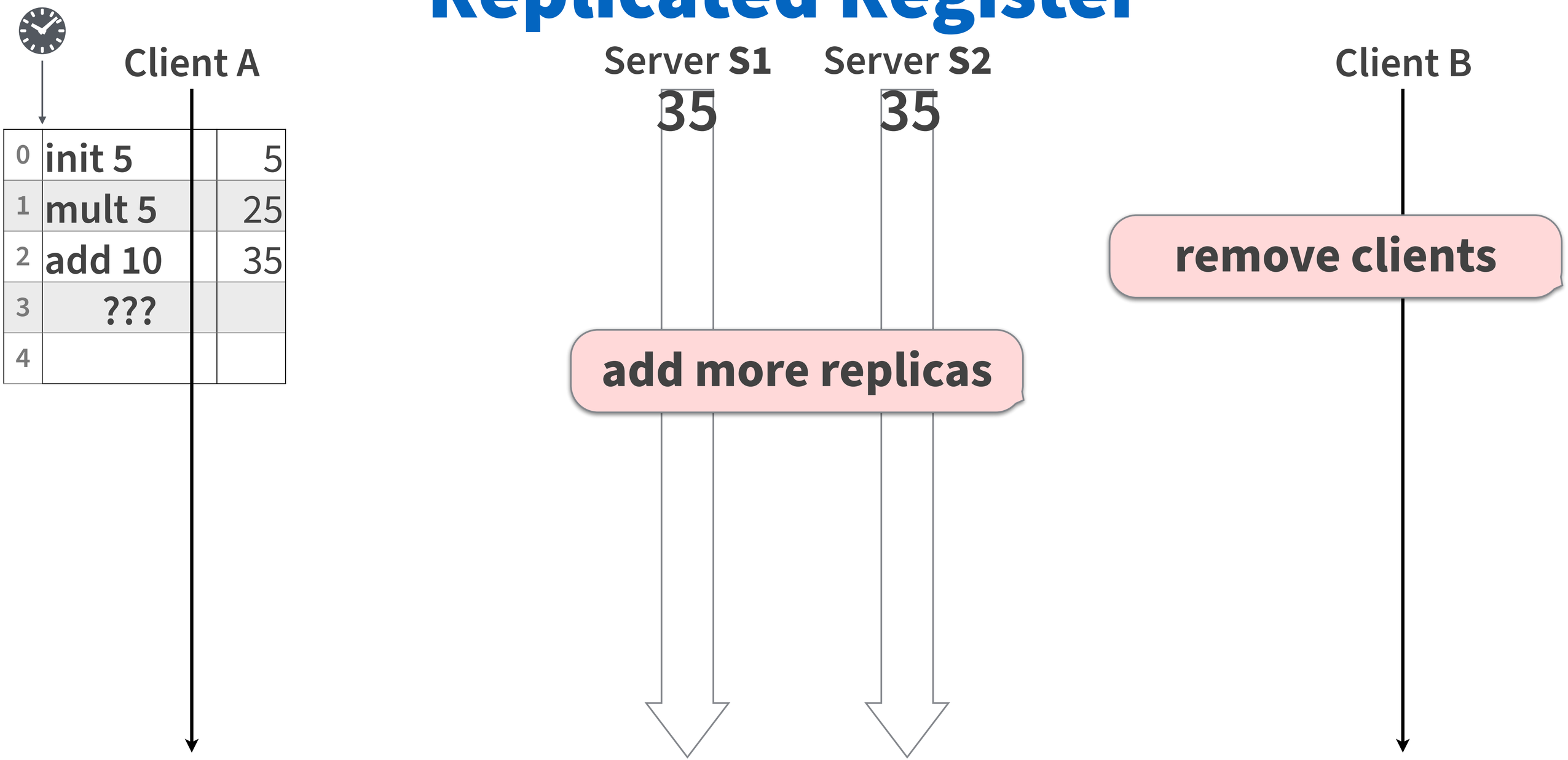
Replicated Register



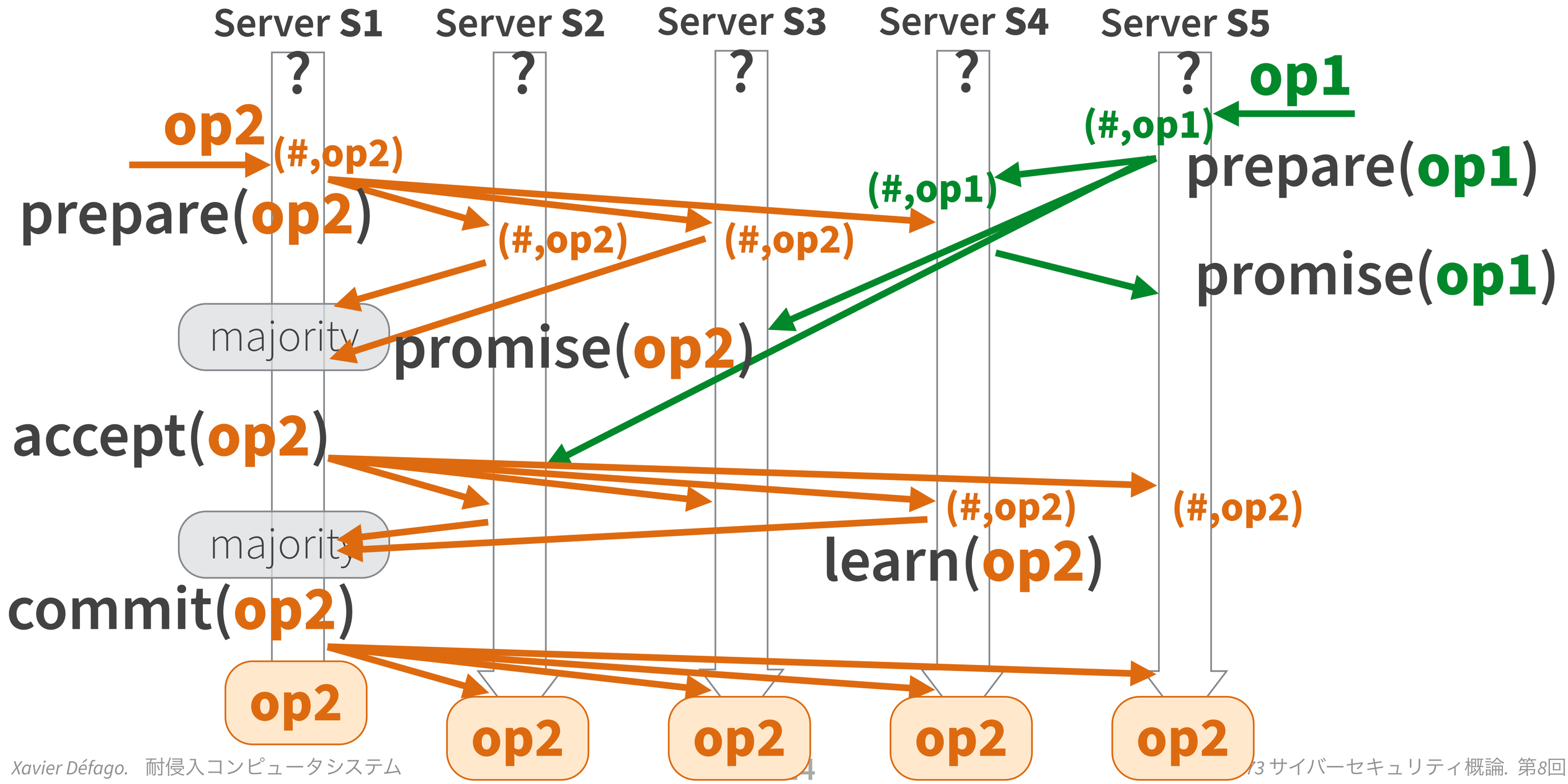
Replicated Register



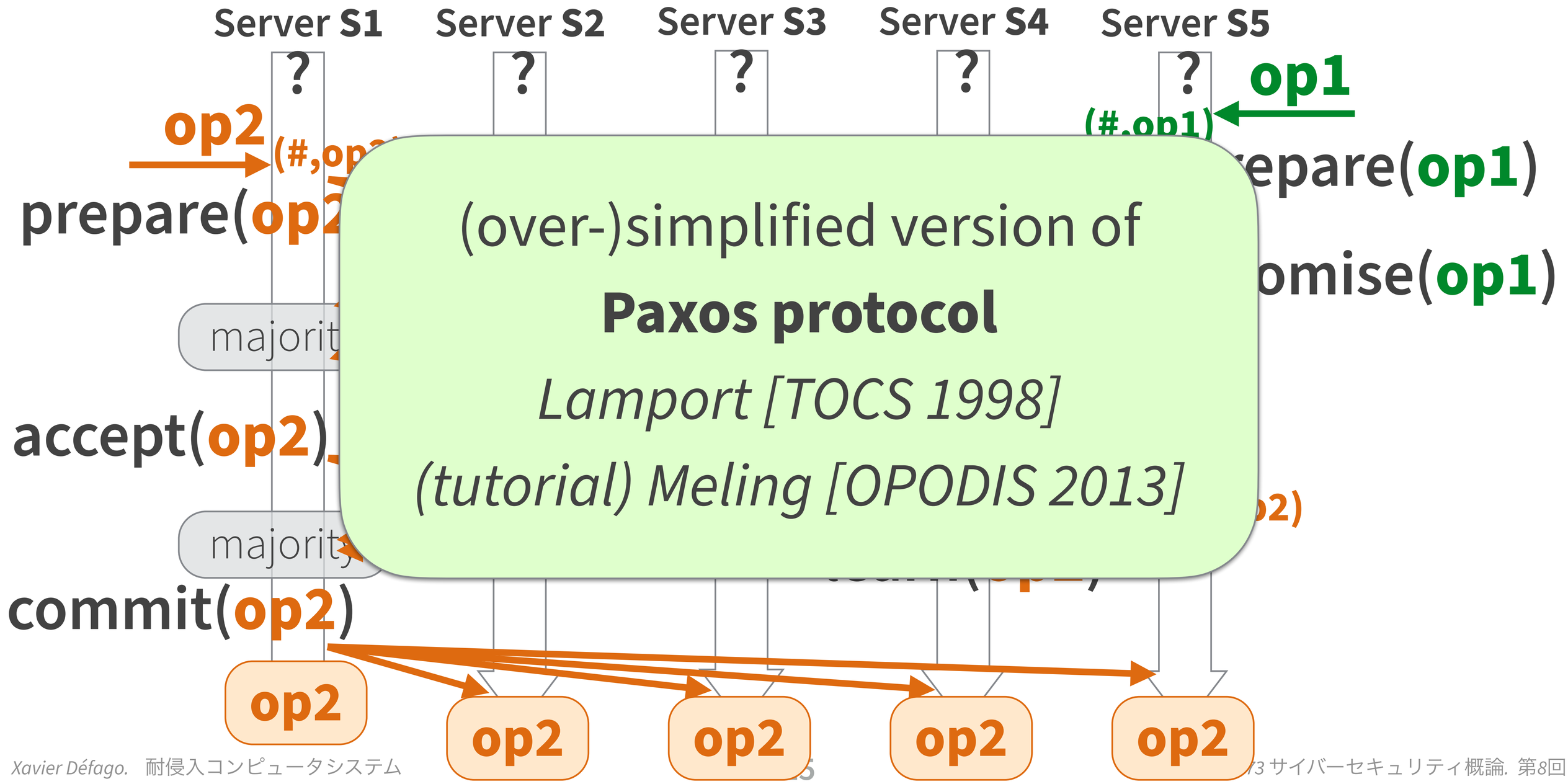
Replicated Register



Replicated Register



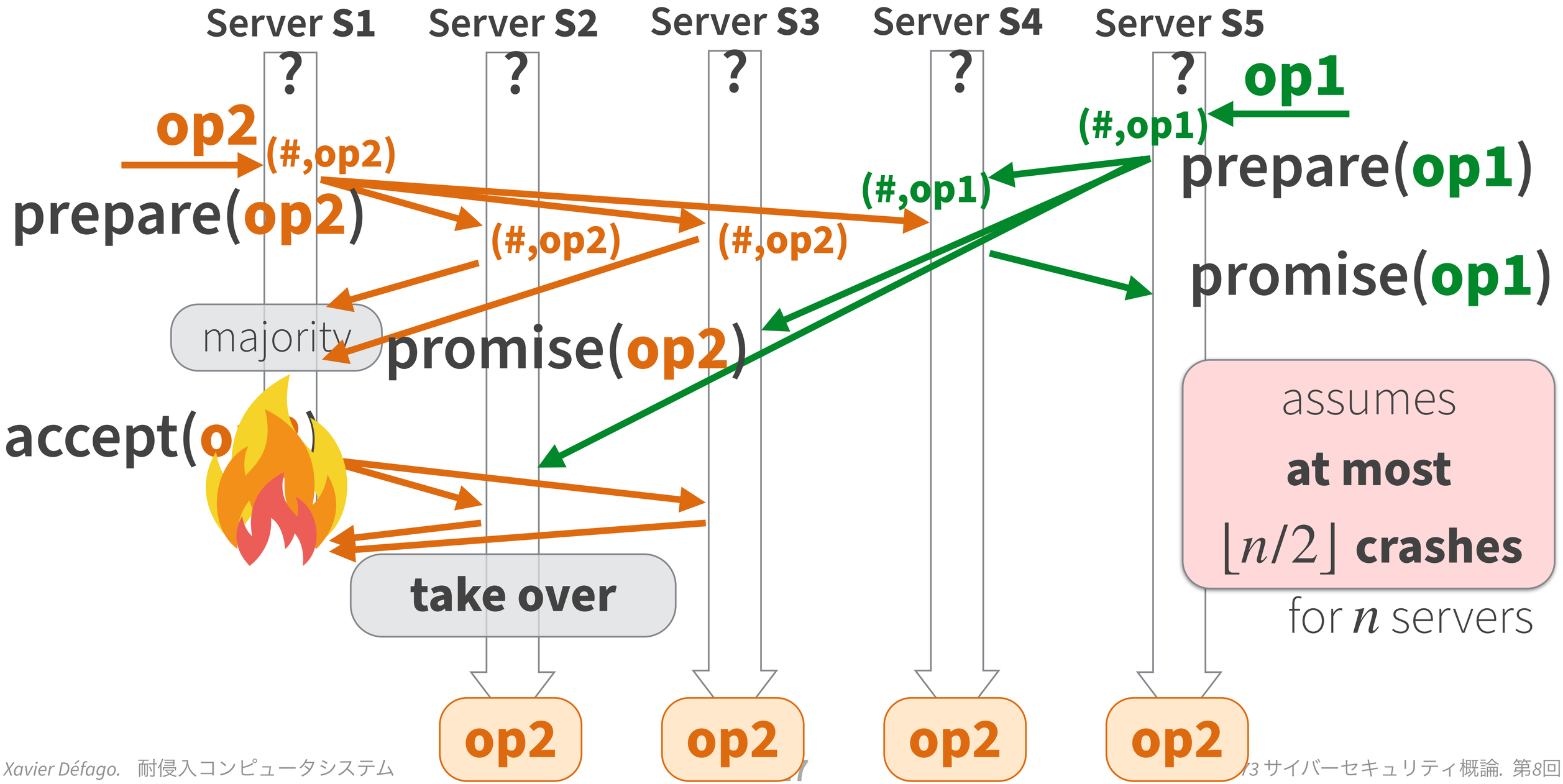
Replicated Register



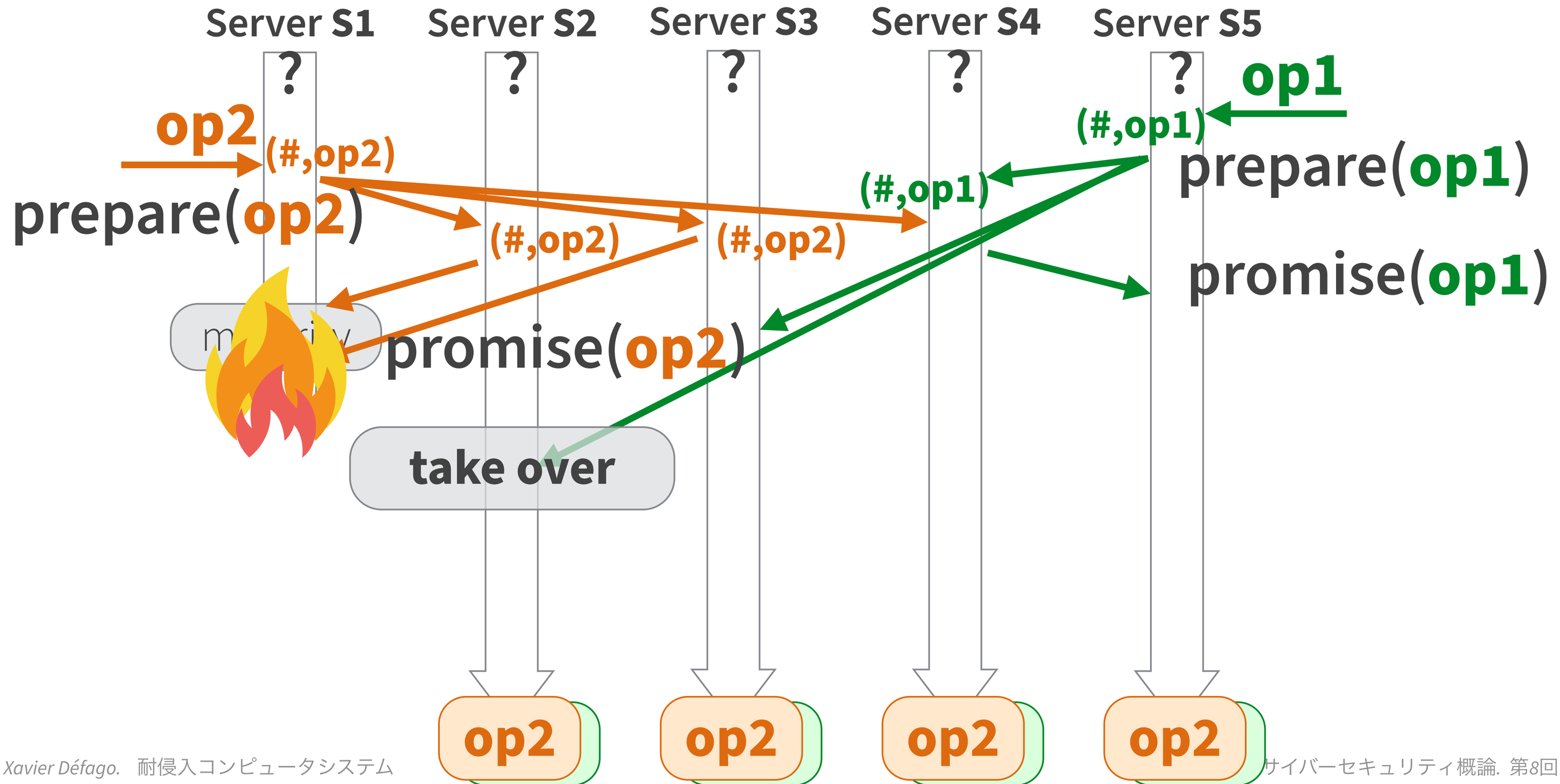
Quorums

Why majority?

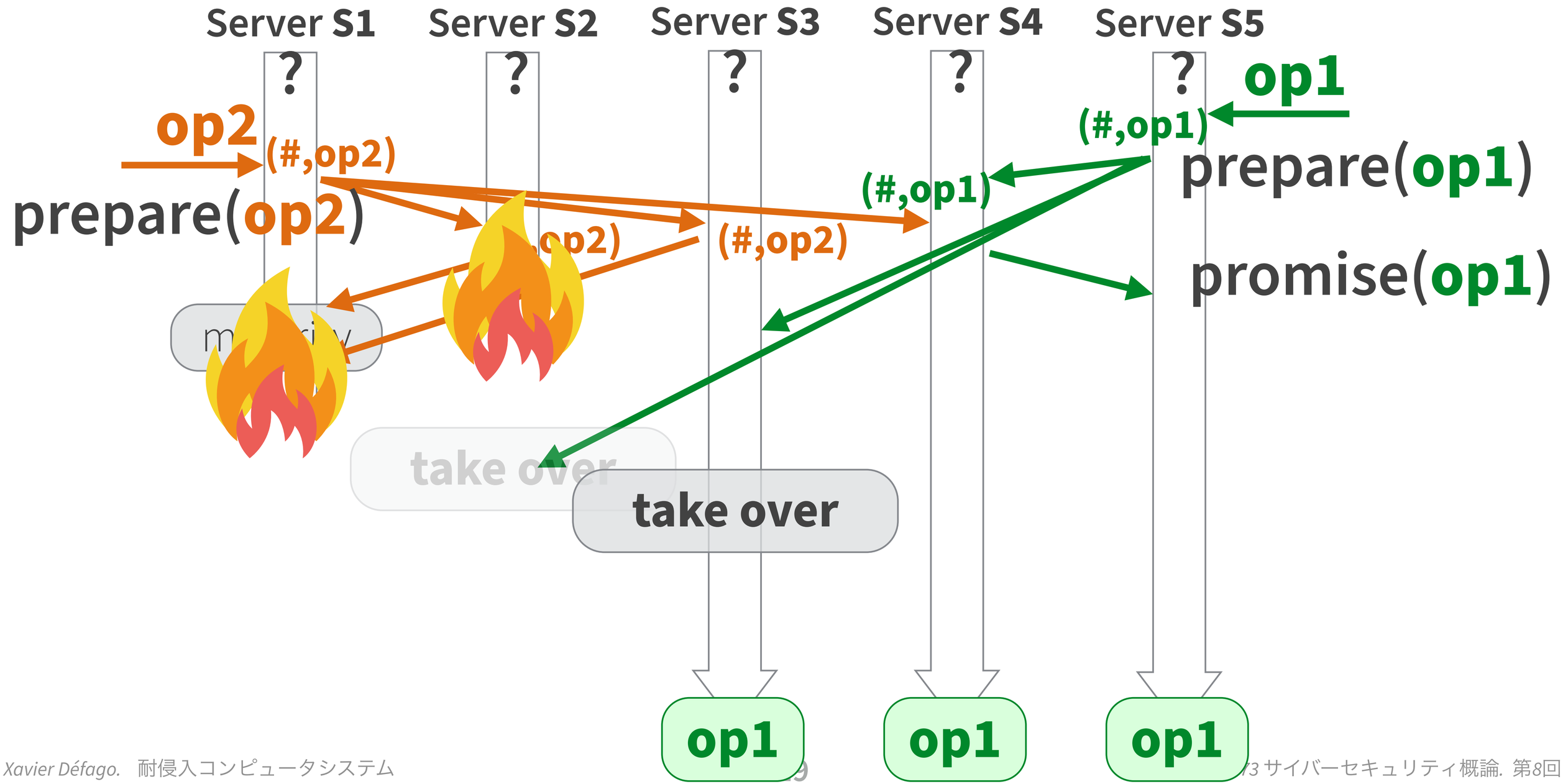
Replicated Register



Replicated Register



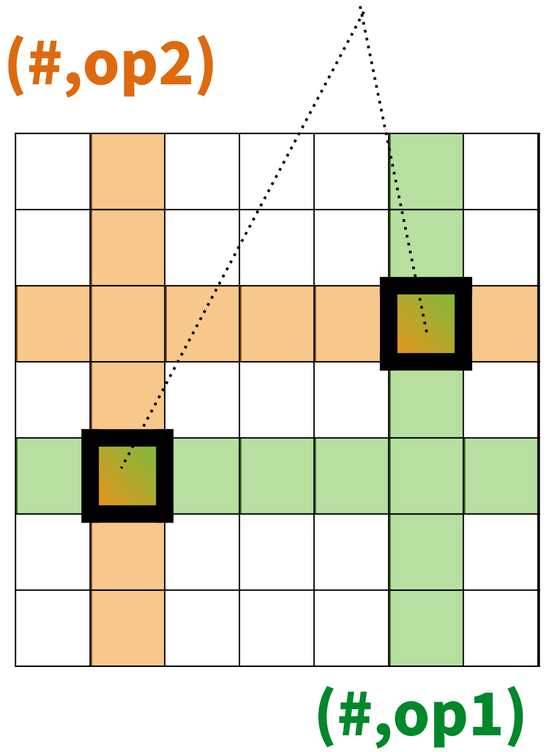
Quorums


Server S1
Server S2
Server S3
Server S4
Server S5
?
?
?
?
?
op2
(#,op2)
prepare(op2)
op1
(#,op1)
prepare(op1)
(#,op1)
(#,op2)
promise(op1)
take over
take over
op1
op1
op1

Quorums

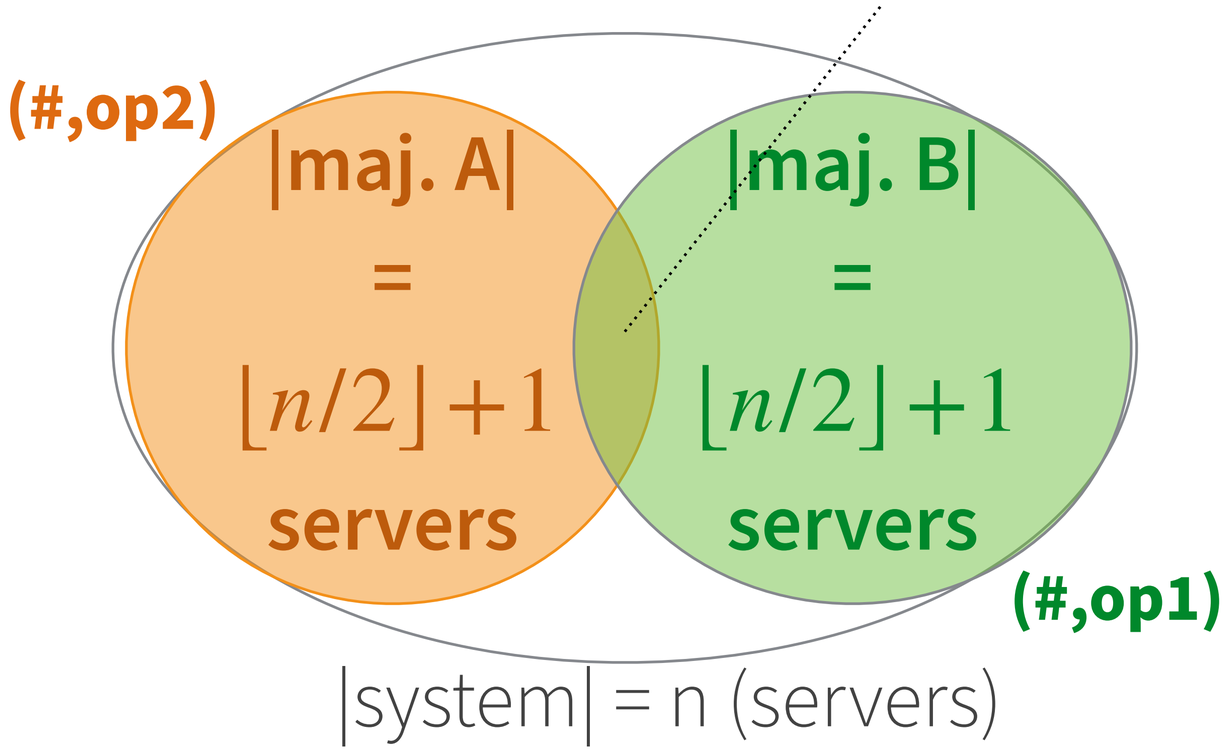
grid quorum

at least two servers



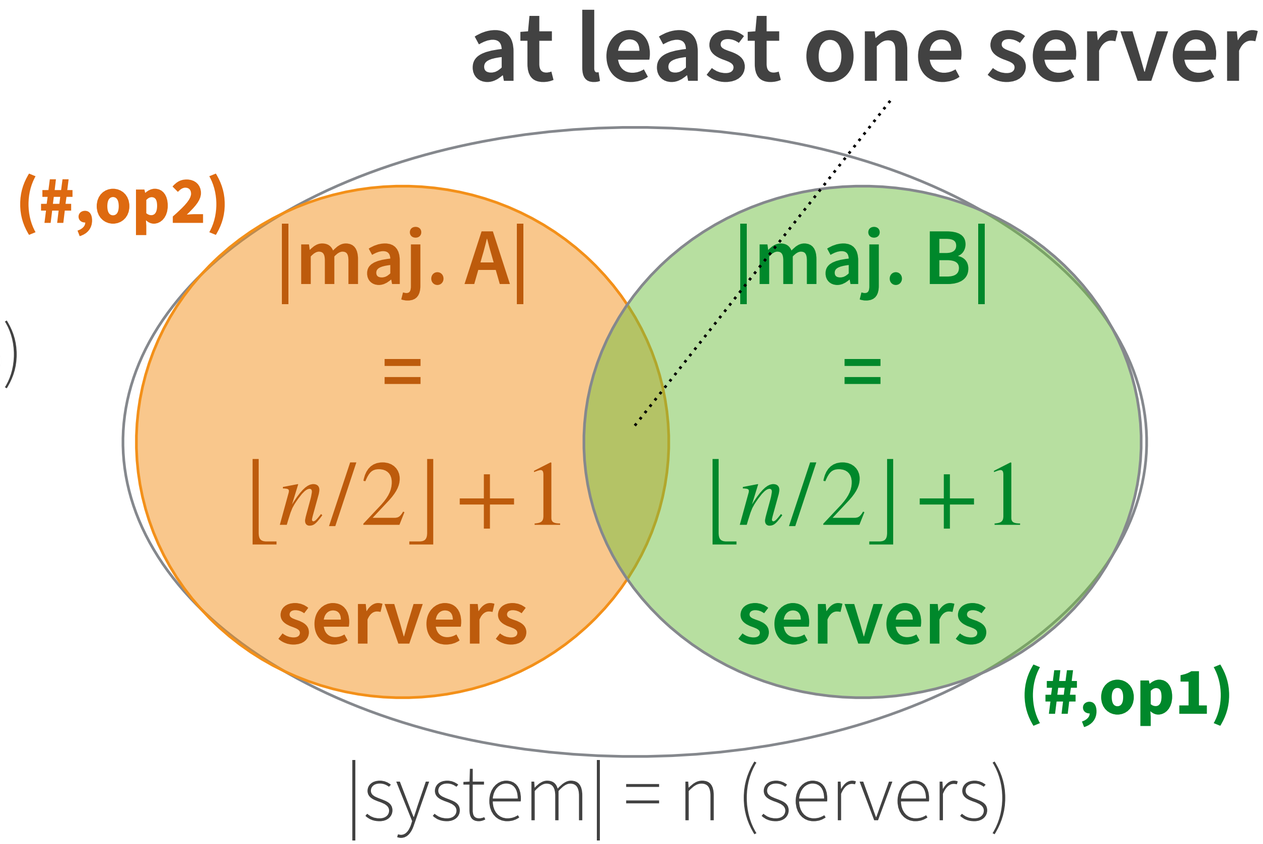
majority quorum

at least one server



Recovery (take over)

- **Terminology**
  - n servers; f faulty (crash); $n \geq 2f+1$
- **Recovery (take over)**
  - accept (#,op2) by majority (A)
  - recovery with info. from majority (B)
  - at least one process from A
- **Non-blocking**
  - broadcast message
  - wait for at most $n-f$ replies



Byzantine Fault-Tolerance

Data / System Integrity

- **Requirements**
  - No tampering of data / system
  - Distributed system: preserve **consistency**
- **“Outside” Threat**
  - Attacker has **no secret** initially
  - Attacker uses **public** service interface in unexpected ways
- **“Inside” Threat**
  - Attacker **passes as legitimate** node / server (eludes detection)
  - Attacker has access to **all information**

Byzantine Faults

- **Fault**
  - behavior deviates from specification
- **Fault Modes**
  - **Crash**

    stops working permanently
  - **Omission**

    misses sending/receiving of messages
  - **Rational (selfish)**

    (mis)behavior that maximizes local benefit
  - **Byzantine** (accidental / malicious)

    any type of misbehavior

Byzantine Faults

Concretely

- broadcasting **<*info*>** as part of an algorithm

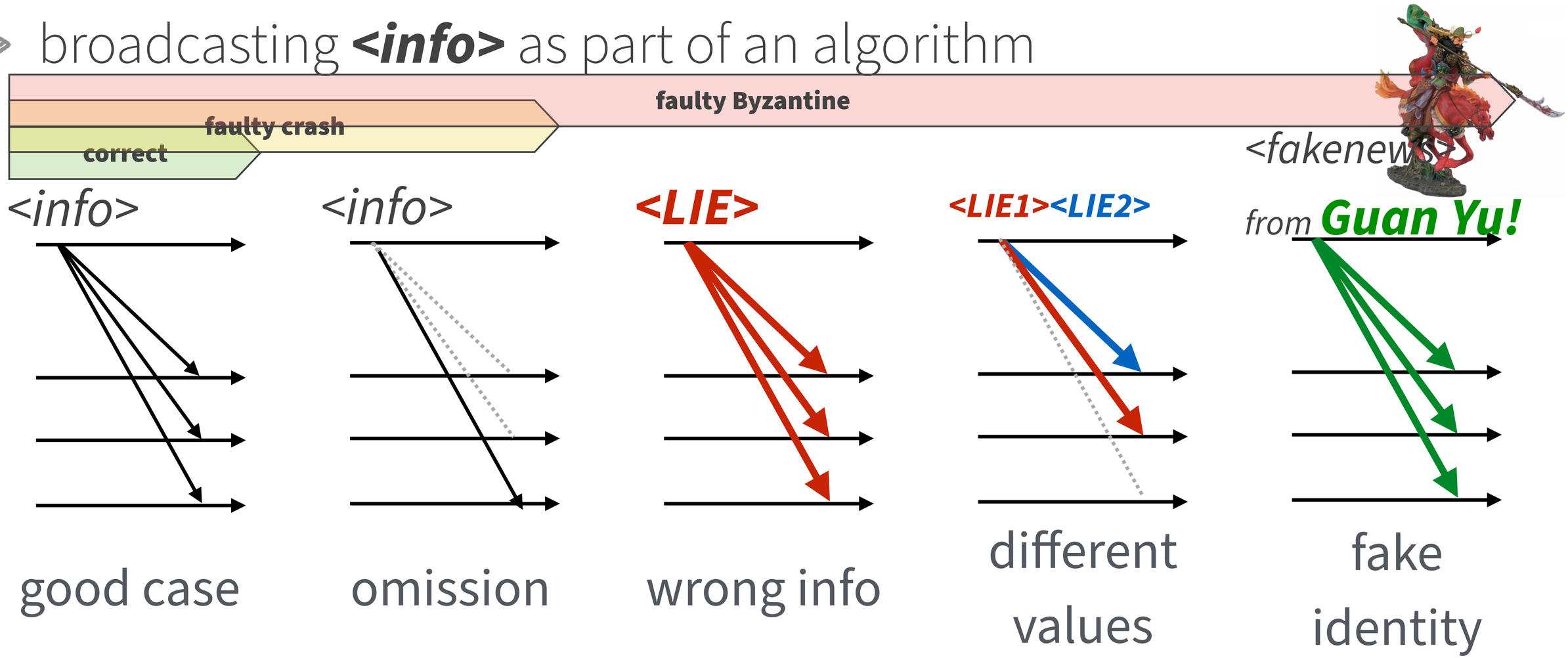
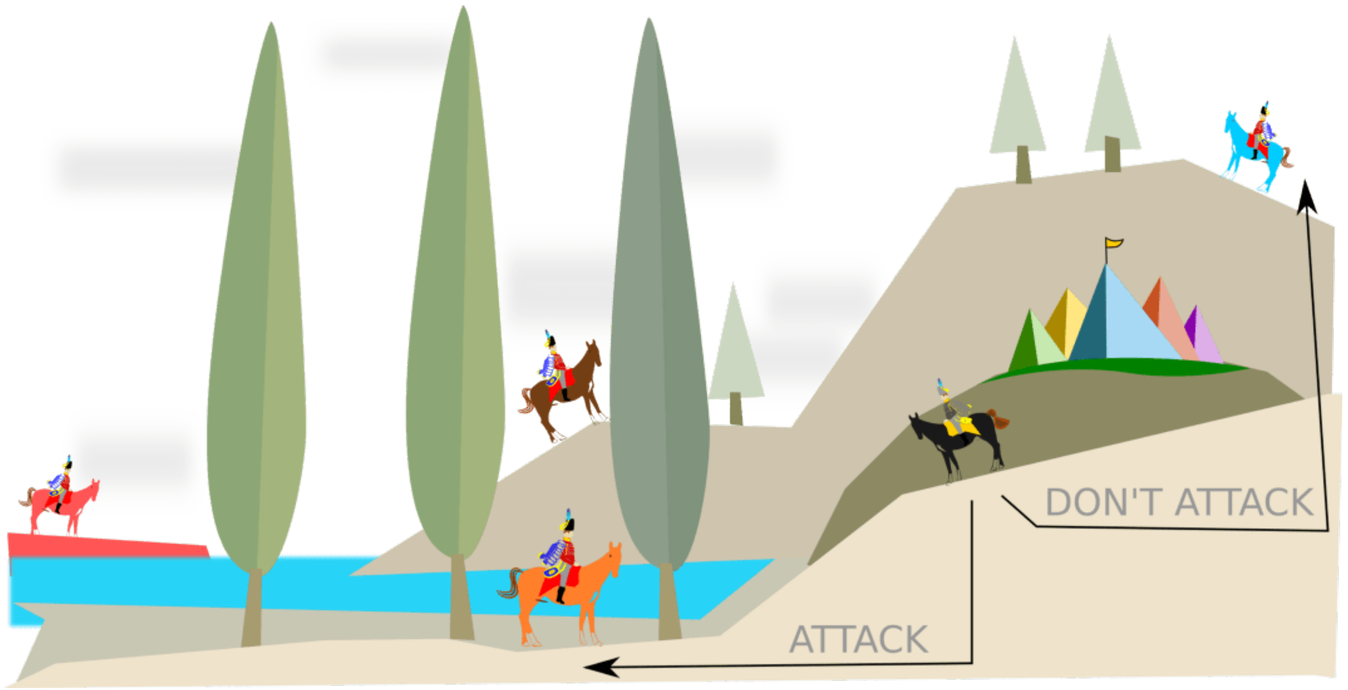


image source: https://www.moss247.com

Byzantine Generals

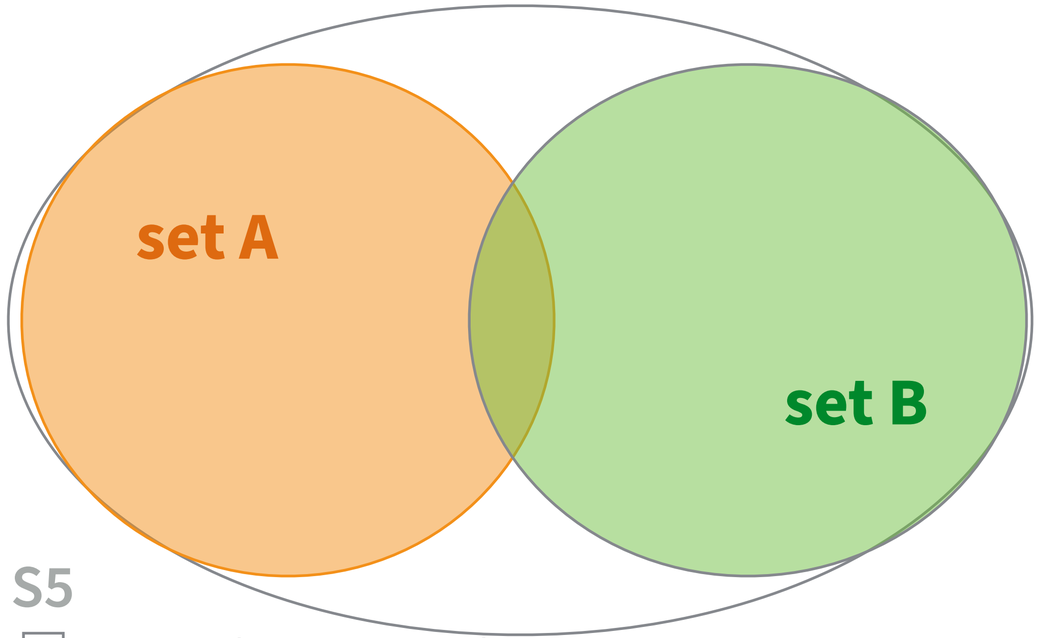


[LSP82] Lamport, Shostak, Pease: **The Byzantine Generals Problem**. *ACM Trans. Program. Lang. Syst.* 4(3): 382-401 (1982)

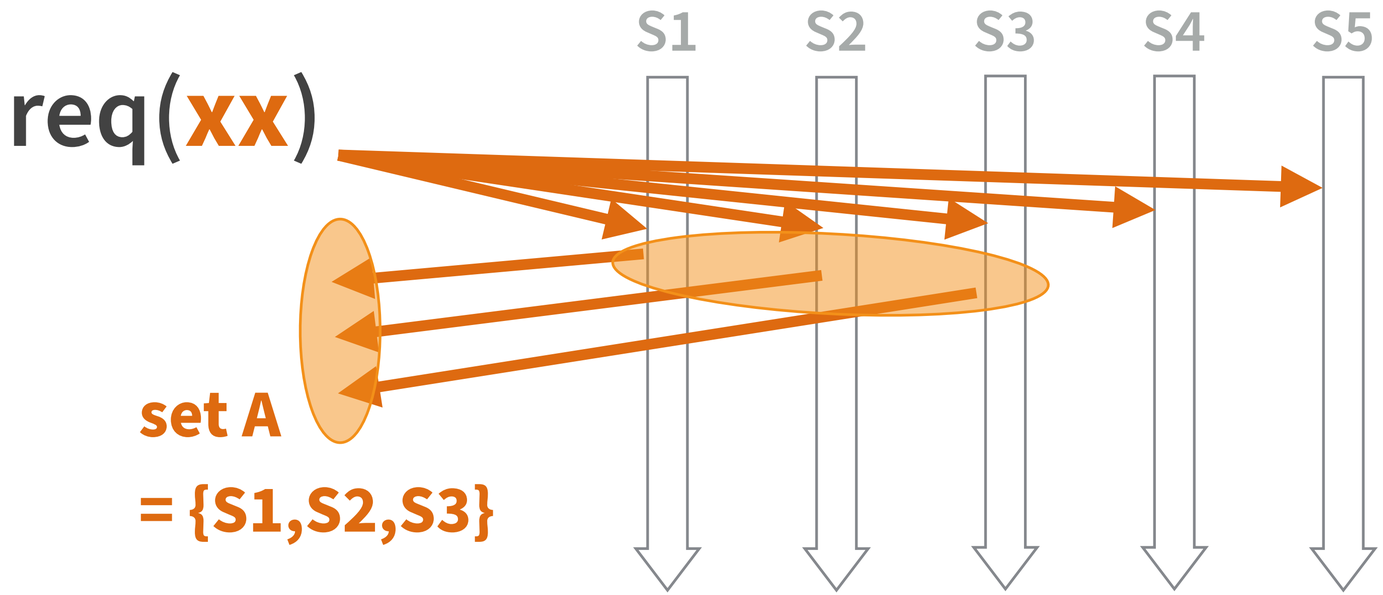
Byzantine Quorum

Terminology

- n servers; $f = f_C + f_B$
- f_C number of **crash** faults
- f_B number of **Byzantine** faults

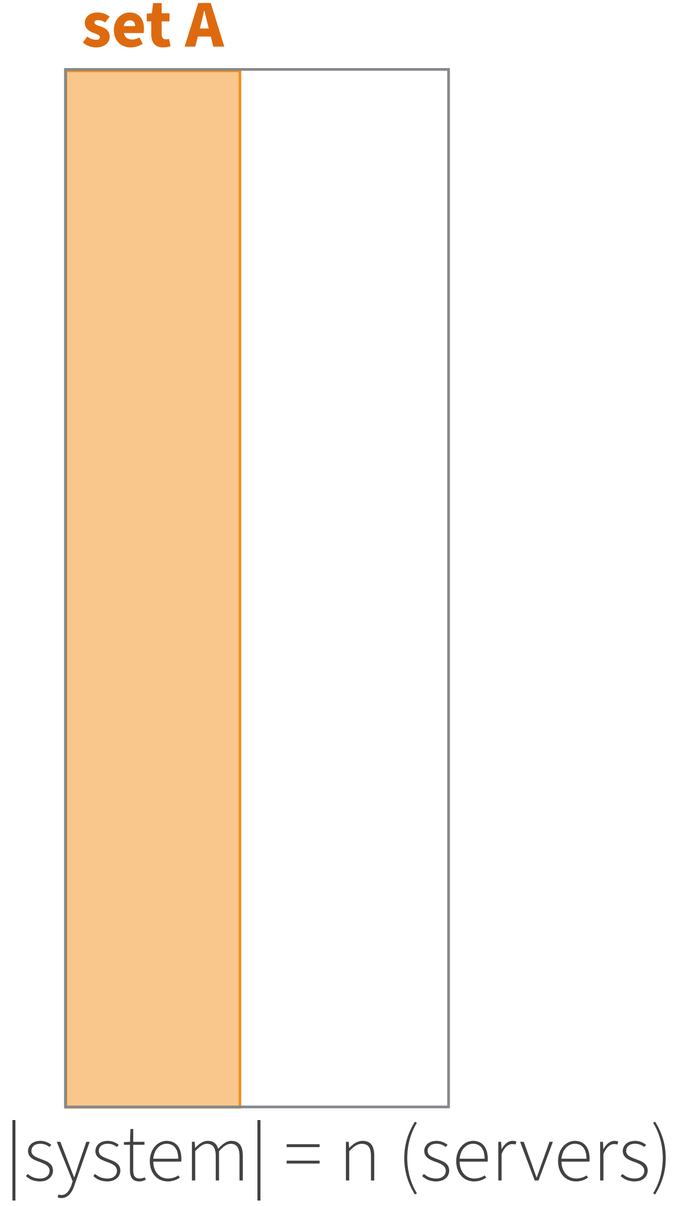


|system| = n (servers)



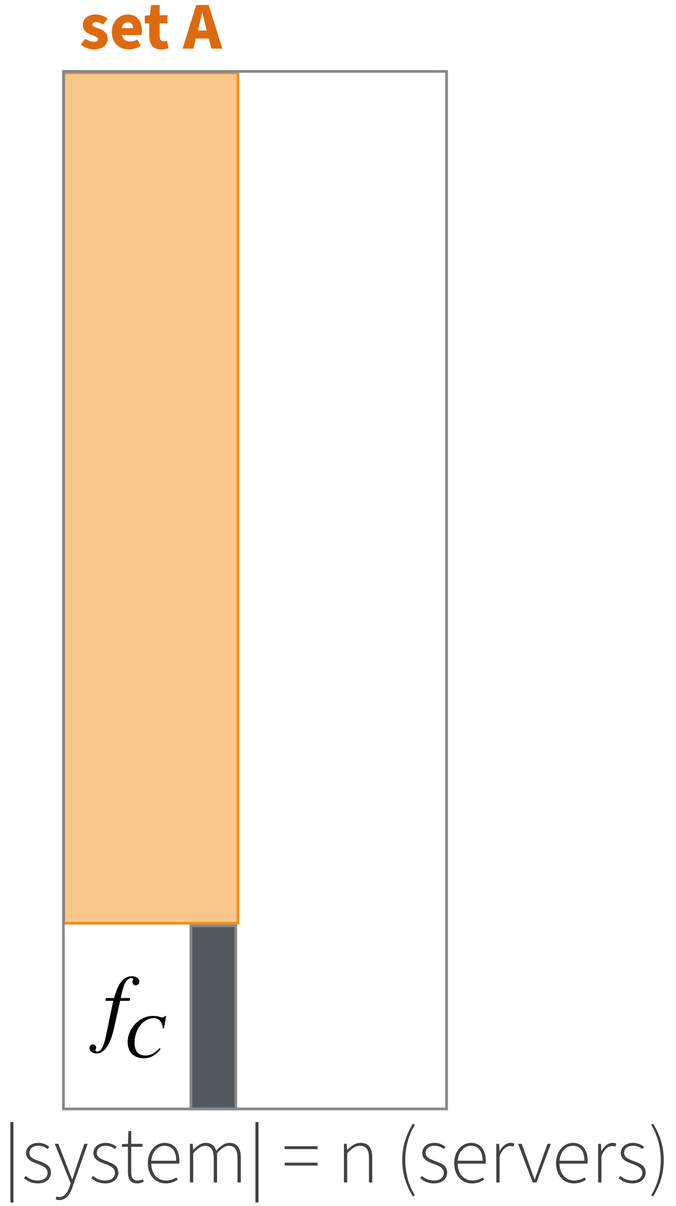
Byzantine Quorum

- **Terminology**
  - n servers; $f = f_C + f_B$
  - f_C number of **crash** faults
  - f_B number of **Byzantine** faults
- **Non-blocking**



Byzantine Quorum

- **Terminology**
  - n servers; $f = f_C + f_B$
  - f_C number of **crash** faults
  - f_B number of **Byzantine** faults
- **Non-blocking**
  - f_C have crashed



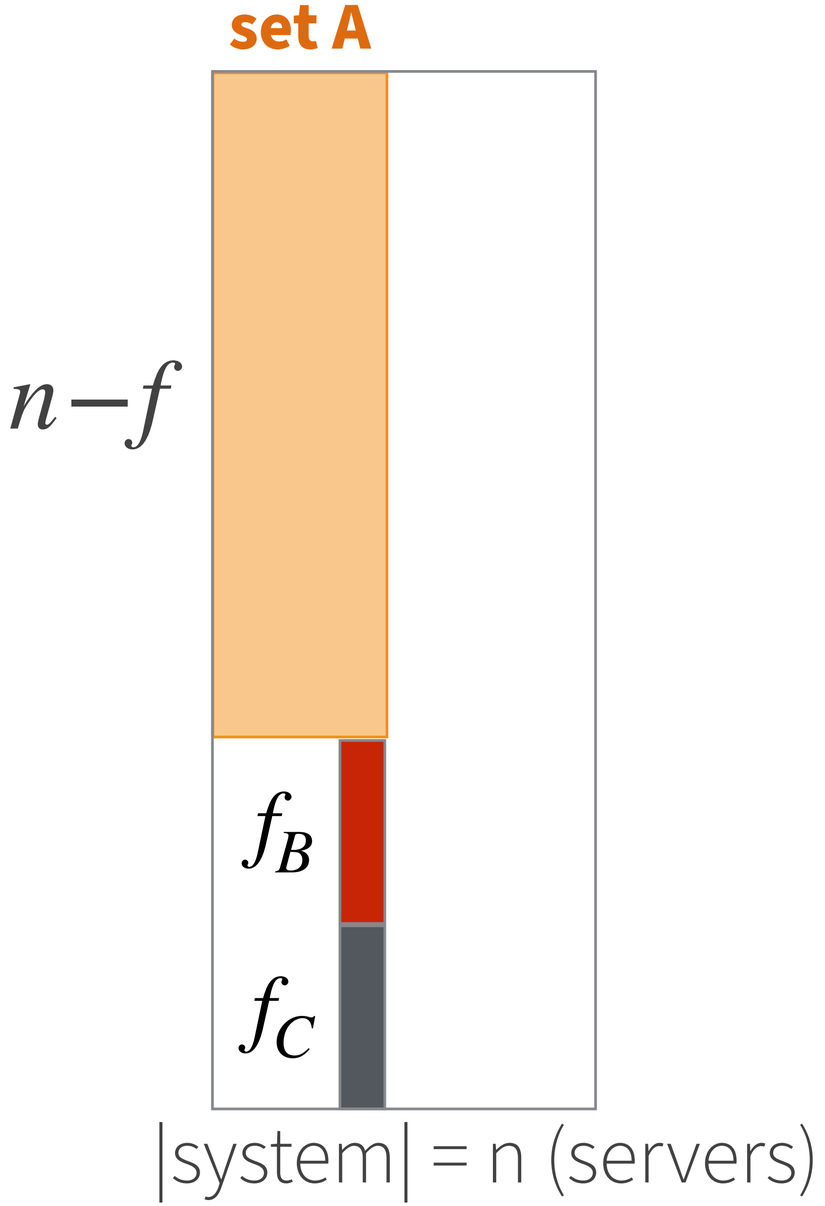
Byzantine Quorum

Terminology

- n servers; $f = f_C + f_B$
- f_C number of **crash** faults
- f_B number of **Byzantine** faults

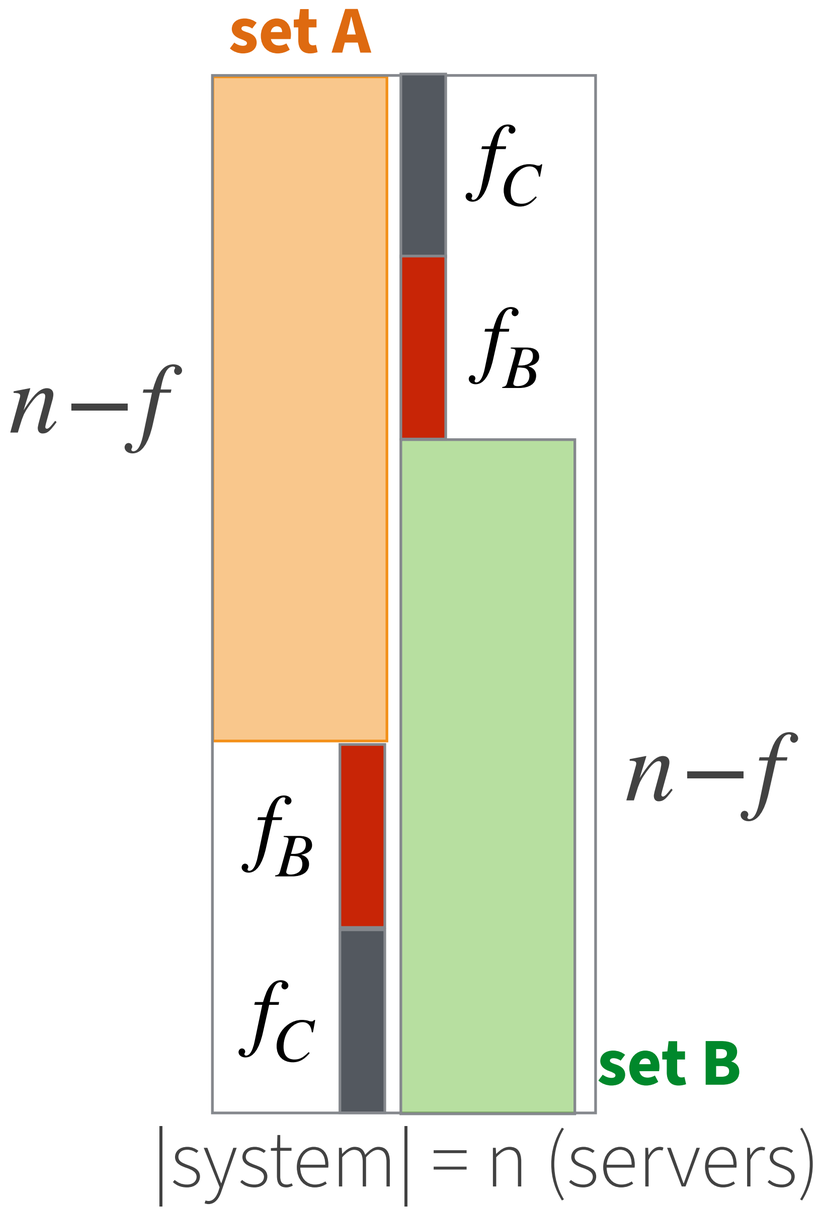
Non-blocking

- f_C have crashed
- f_B never reply



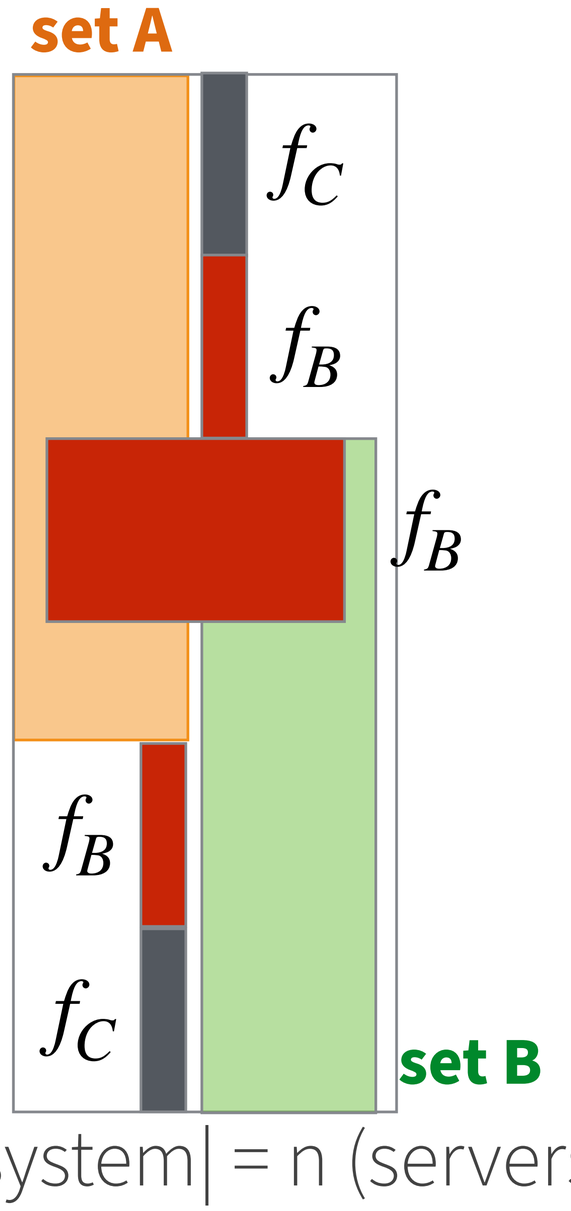
Byzantine Quorum

- **Terminology**
  - n servers; $f = f_C + f_B$
  - f_C number of **crash** faults
  - f_B number of **Byzantine** faults
- **Non-blocking**
  - f_C have crashed
  - f_B never reply
- **Intersection**



Byzantine Quorum

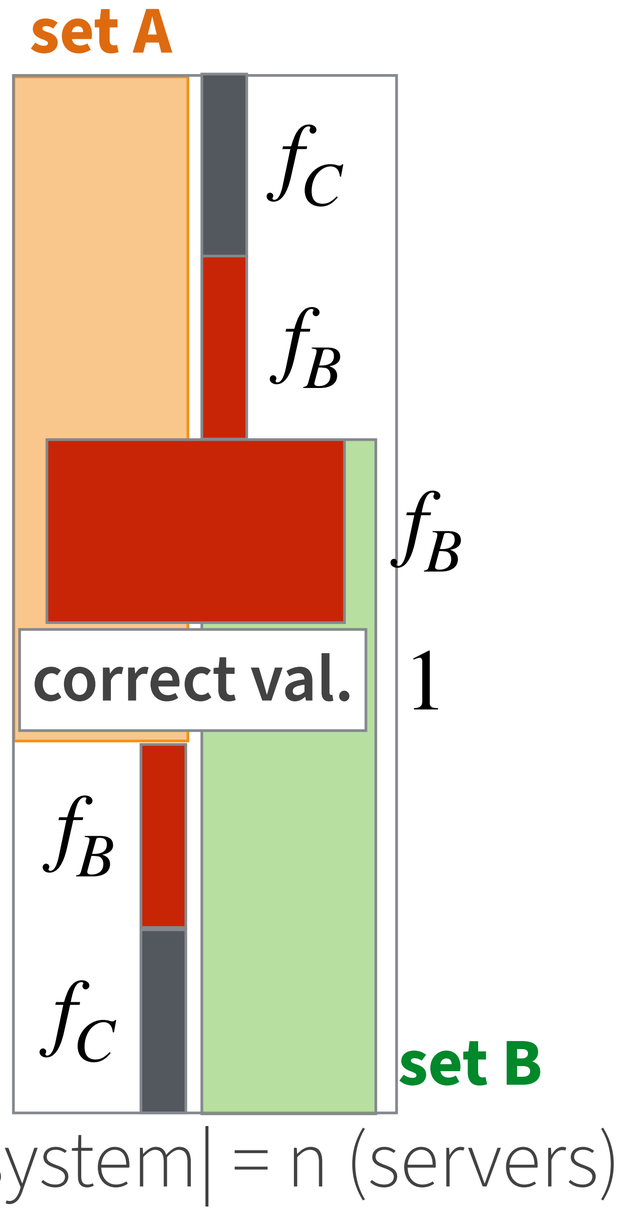
- **Terminology**
  - n servers; $f = f_C + f_B$
  - f_C number of **crash** faults
  - f_B number of **Byzantine** faults
- **Non-blocking**
  - f_C have crashed
  - f_B never reply
- **Intersection**
  - includes lies



|system| = n (servers)

Byzantine Quorum

- **Terminology**
  - n servers; $f = f_C + f_B$
  - f_C number of **crash** faults
  - f_B number of **Byzantine** faults
- **Non-blocking**
  - f_C have crashed
  - f_B never reply
- **Intersection**
  - includes lies; at least one correct value

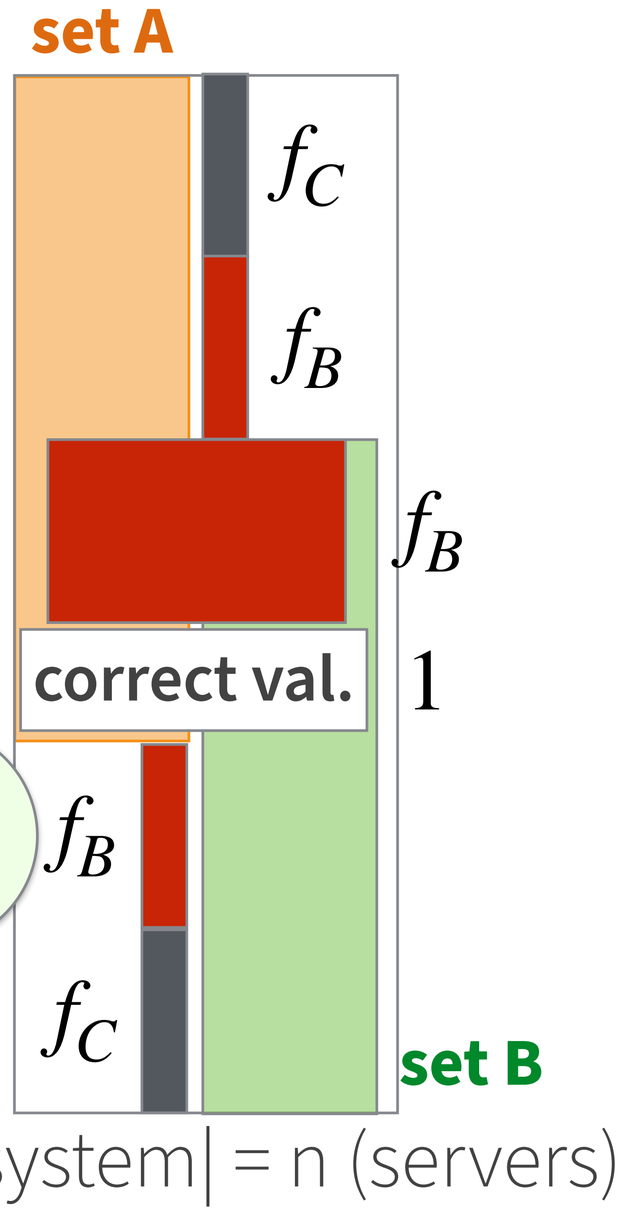


|system| = n (servers)

Byzantine Quorum

- **Terminology**
  - n servers; $f = f_C + f_B$
  - f_C number of **crash** faults
  - f_B number of **Byzantine** faults
- **Non-blocking**
  - f_C have crashed
  - f_B never reply
- **Intersection**
  - includes lies; at least one correct value

$$n \geq 2f_C + 3f_B + 1$$



Practical Byzantine Fault Tolerance

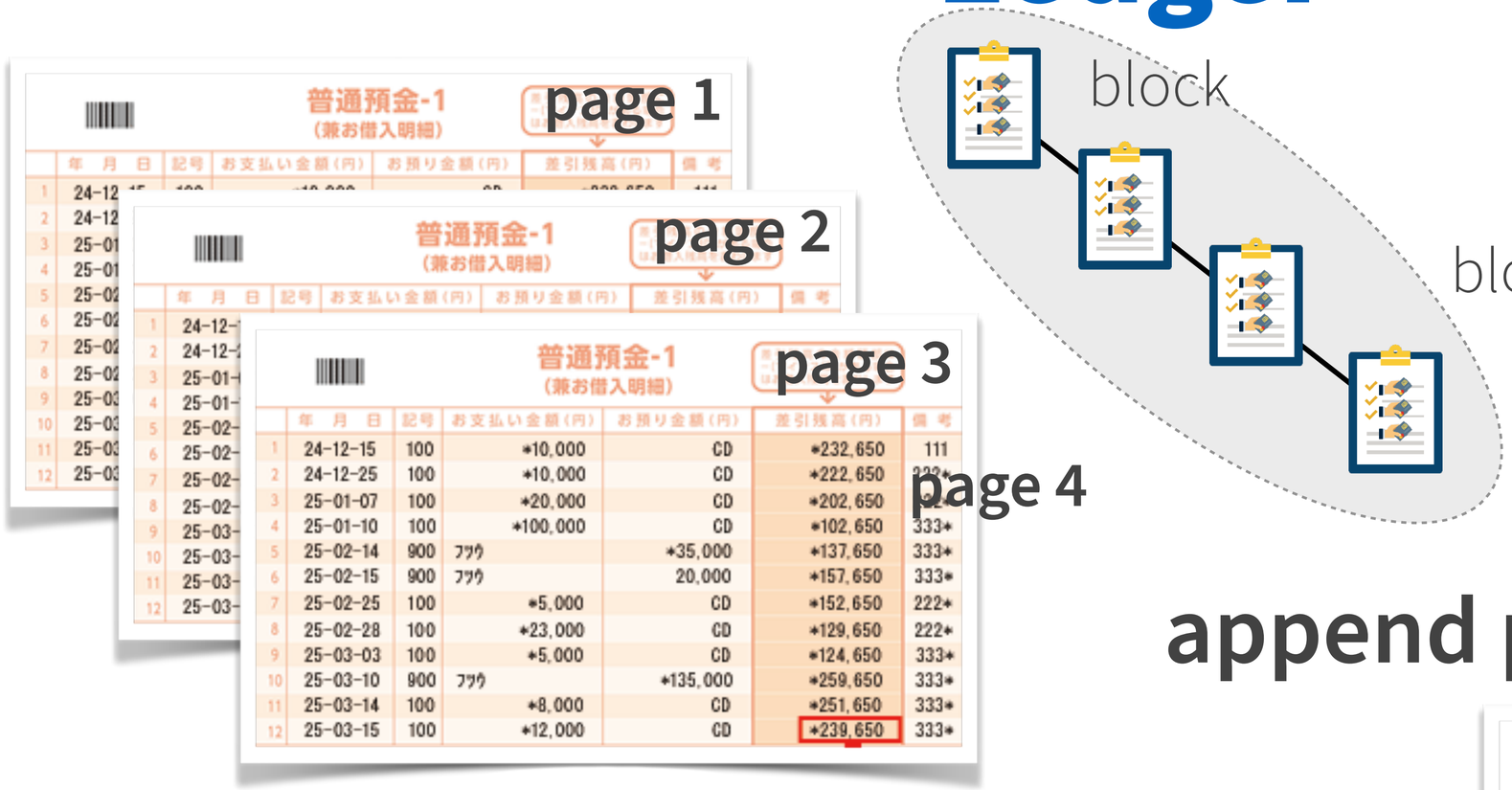
Context

- described in client/server model
- state-machine replication
- similarities with Paxos
- Byzantine-tolerant
- uses crypto hash function

[CL01] Castro, Liskov: **Practical Byzantine Fault-Tolerance and Proactive Recovery**. *ACM Trans. Comput. Syst.* **20**(4): 398-461 (2001)

Blockchain

Ledger



普通預金-1
（兼お借入明細）

	年 月 日	記号	お支払い金額（円）	お預り金額（円）	差引残高（円）	備 考
1	24-12-15	100	⋆10,000	CD	⋆232,650	111
2	24-12-25	100	⋆10,000	CD	⋆222,650	222⋆
3	25-01-07	100	⋆20,000	CD	⋆202,650	222⋆
4	25-01-10	100	⋆100,000	CD	⋆102,650	333⋆
5	25-02-14	900	ﾌﾘｳ	⋆35,000	⋆137,650	333⋆
6	25-02-15	900	ﾌﾘｳ	20,000	⋆157,650	333⋆
7	25-02-25	100	⋆5,000	CD	⋆152,650	222⋆
8	25-02-28	100	⋆23,000	CD	⋆129,650	222⋆
9	25-03-03	100	⋆5,000	CD	⋆124,650	333⋆
10	25-03-10	900	ﾌﾘｳ	⋆135,000	⋆259,650	333⋆
11	25-03-14	100	⋆8,000	CD	⋆251,650	333⋆
12	25-03-15	100	⋆12,000	CD	⋆239,650	333⋆

append page

普通預金-1
（兼お借入明細）

	年 月 日	記号	お支払い金額（円）	お預り金額（円）	差引残高（円）	備 考
1	24-12-15	100	⋆10,000	CD	⋆232,650	111
2	24-12-25	100	⋆10,000	CD	⋆222,650	222⋆
3	25-01-07	100	⋆20,000	CD	⋆202,650	222⋆
4	25-01-10	100	⋆100,000	CD	⋆102,650	333⋆
5	25-02-14	900	ﾌﾘｳ	⋆35,000	⋆137,650	333⋆
6	25-02-15	900	ﾌﾘｳ	20,000	⋆157,650	333⋆
7	25-02-25	100	⋆5,000	CD	⋆152,650	222⋆
8	25-02-28	100	⋆23,000	CD	⋆129,650	222⋆
9	25-03-03	100	⋆5,000	CD	⋆124,650	333⋆
10	25-03-10	900	ﾌﾘｳ	⋆135,000	⋆259,650	333⋆
11	25-03-14	100	⋆8,000	CD	⋆251,650	333⋆
12	25-03-15	100	⋆12,000	CD	⋆239,650	333⋆

Distributed Ledger

assumed final/stable
& agreed upon

new block

Root

append to
block with

sybil attack
create many identities
overcome reputation system
circumvent incentive mechanism

finality
not guaranteed

very unlikely in practice

arbitrary depth

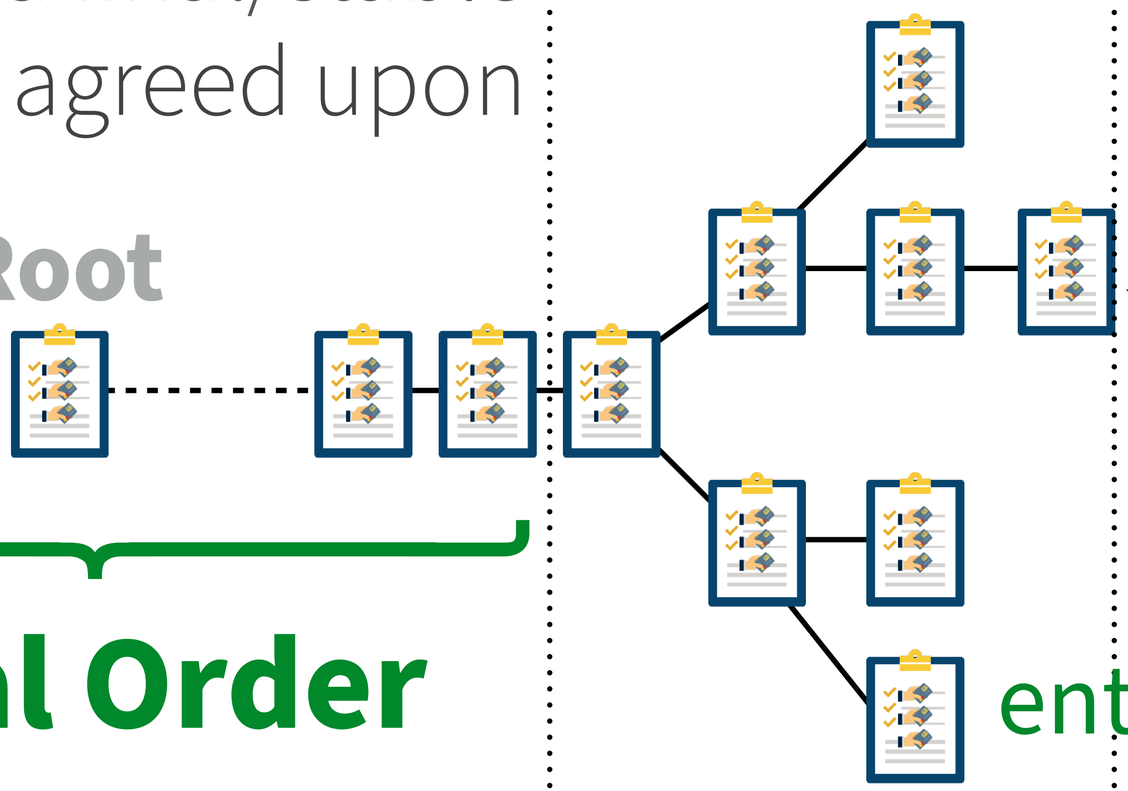
(typ. 5-6 blocks for Bitcoin)

Distributed Ledger

assumed final/stable
& agreed upon

new block

Root



append to
block with
best "survival"
chance

Total Order

on operations

entry := **operation**

arbitrary depth

(typ. 5-6 blocks for Bitcoin)

Comparison

- **Byzantine fault-tolerance**
  - strong consistency
  - **always safe**; mostly live
  - **partition**: majority-only
  - finality: commit => permanent
- **Blockchain**
  - mostly safe; **always live**
  - **partition**: loss at merge
  - **finality**: high probability

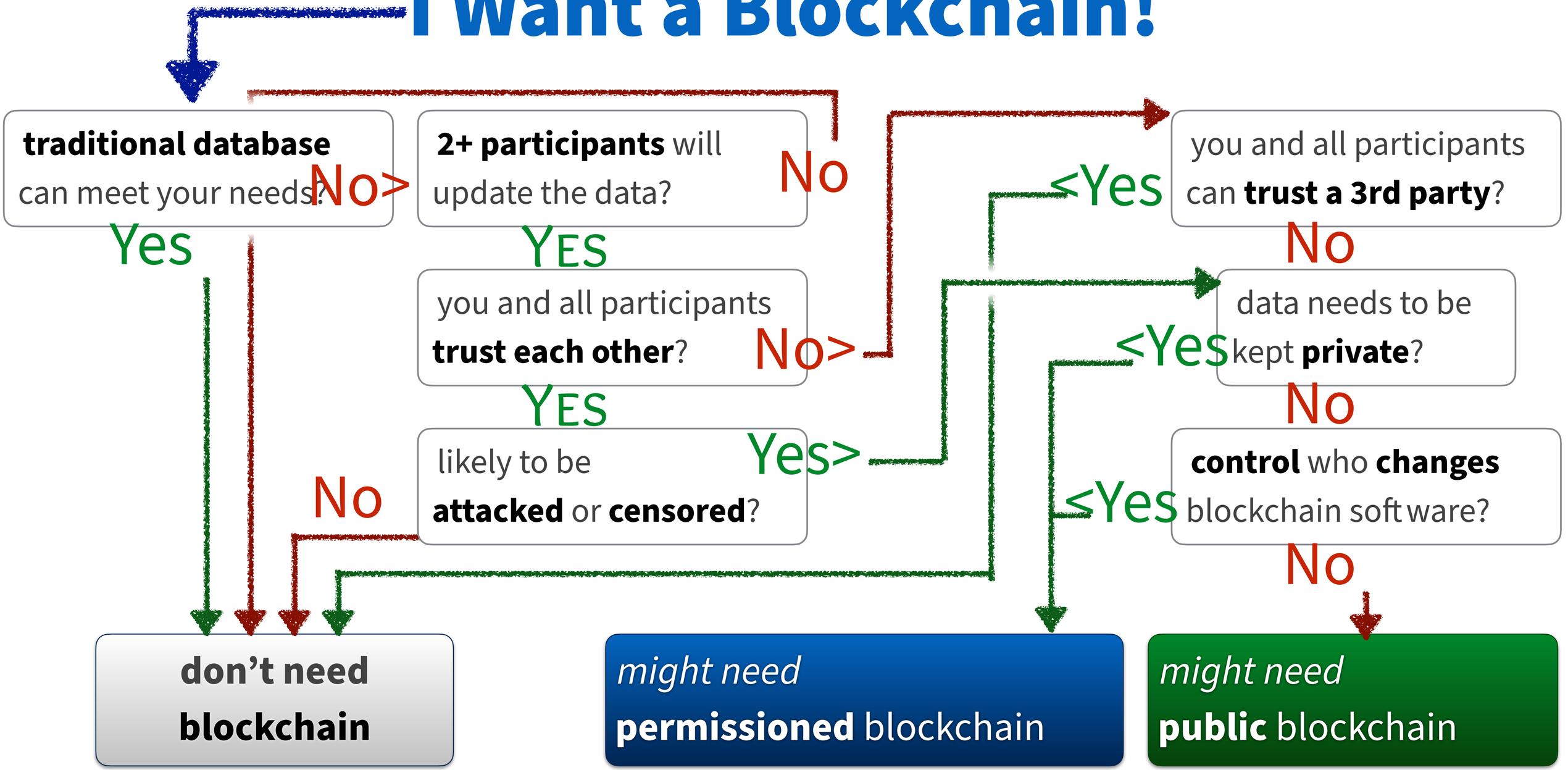
- **Overall**
  - each serves purpose
  - address different issues
  - not mutually exclusive
  - BFT used in blockchain (PoS)
  - tradeoff: CAP theorem

    consistency

    availability

    partition-tolerance

I Want a Blockchain!



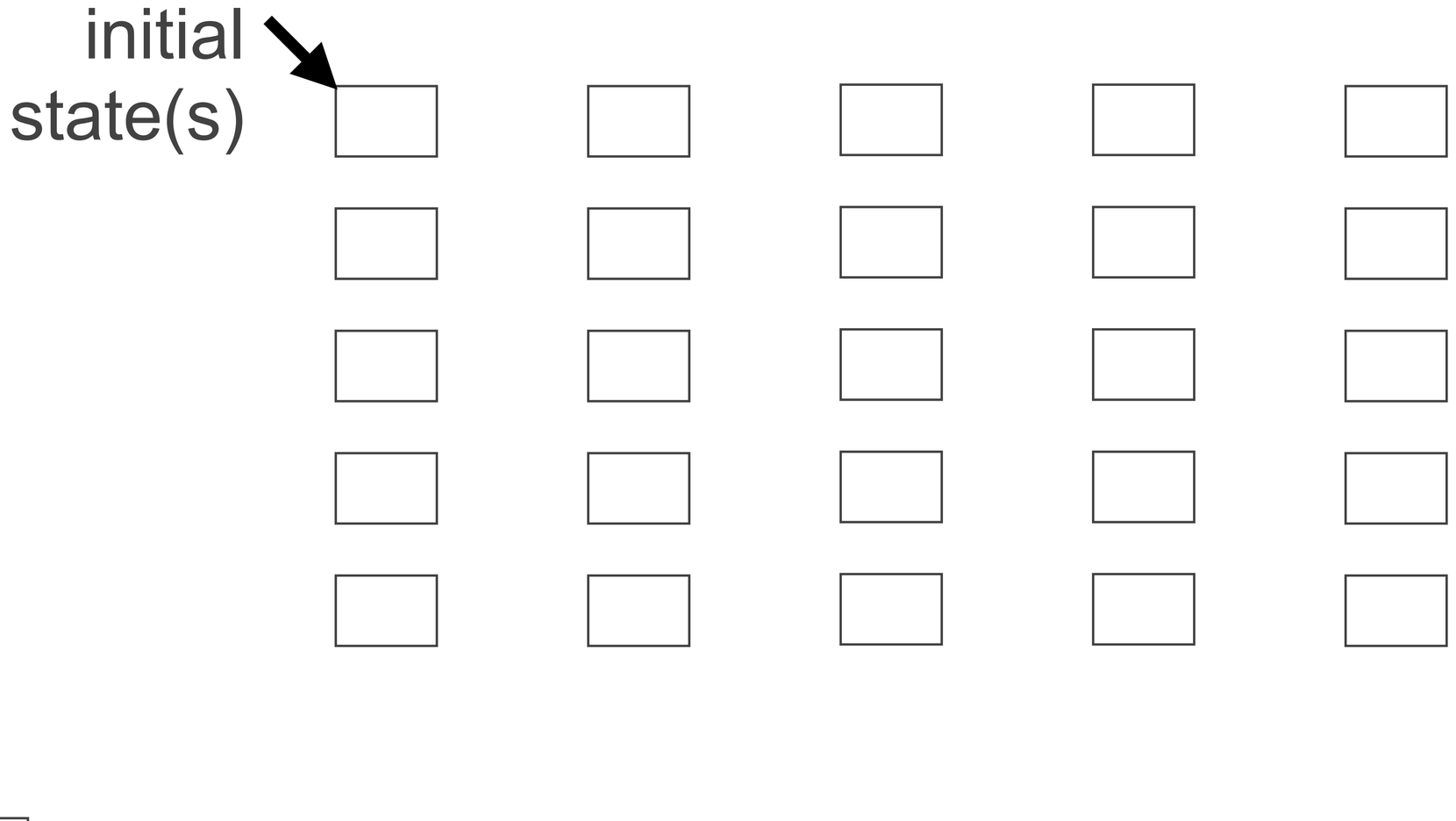
Source: M.E.Peck. Do you need a blockchain? *IEEE Spectrum*, 54(10):38-39, Oct 2017.

Summing Up

- **Distributed Ledger**
  - state-machine replication
  - quorums, agreement
  - problem equivalence
- **Byzantine fault-tolerance**
  - Byzantine faults
  - strong consistency
  - finality
- **Blockchain**
  - distributed ledger
  - eventual consistency
  - no finality

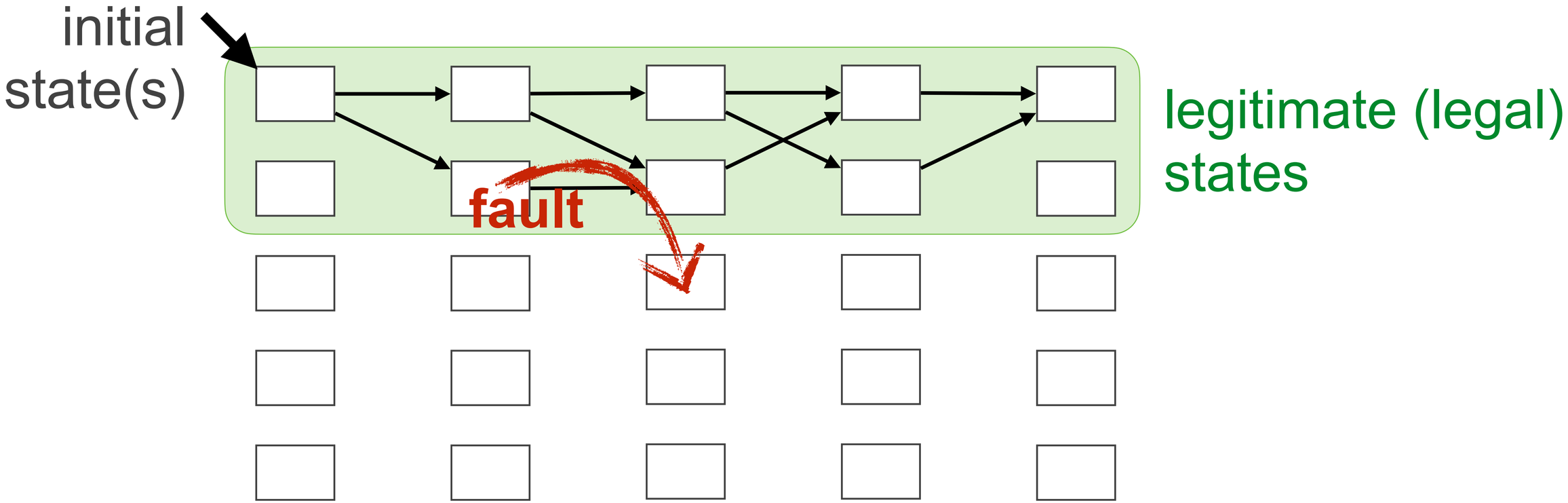
Self-Stabilization

Systems



system state

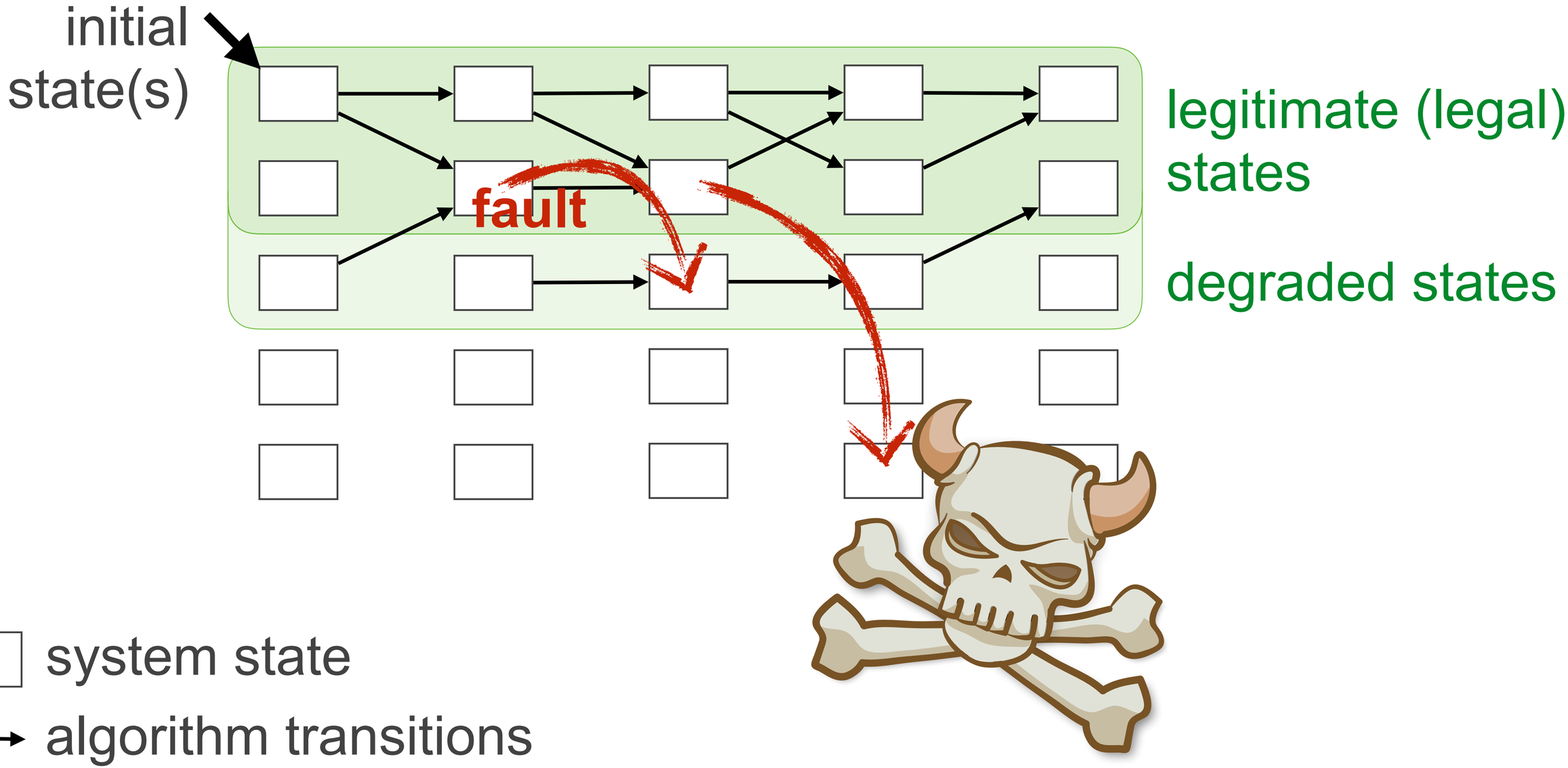
Systems



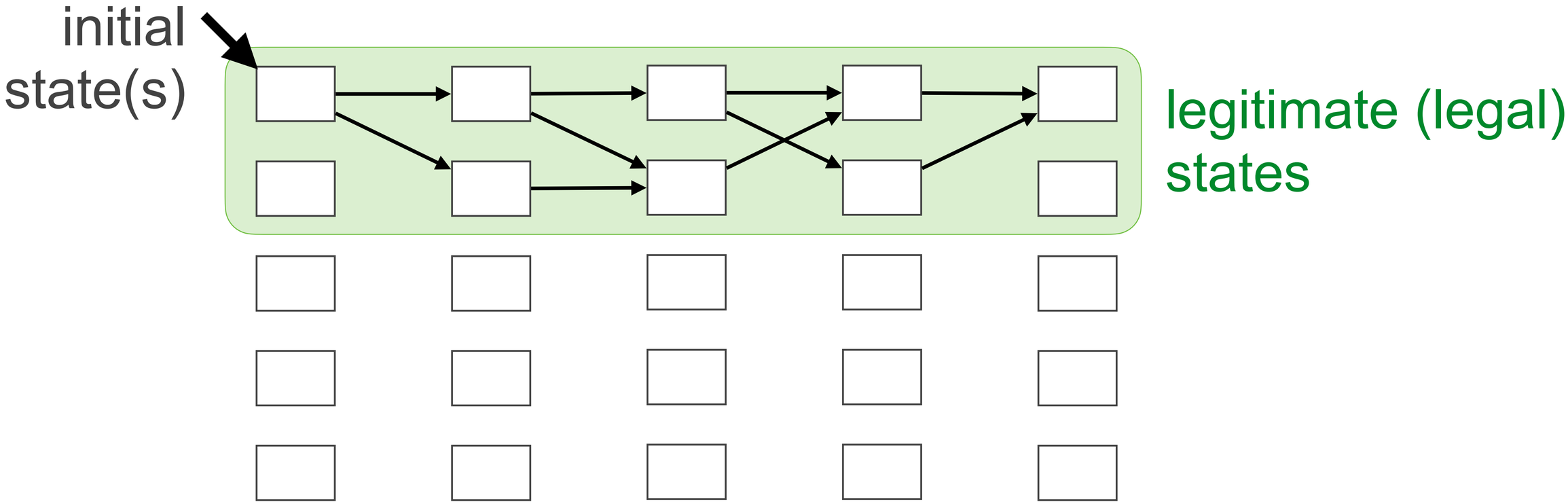
system state

algorithm transitions

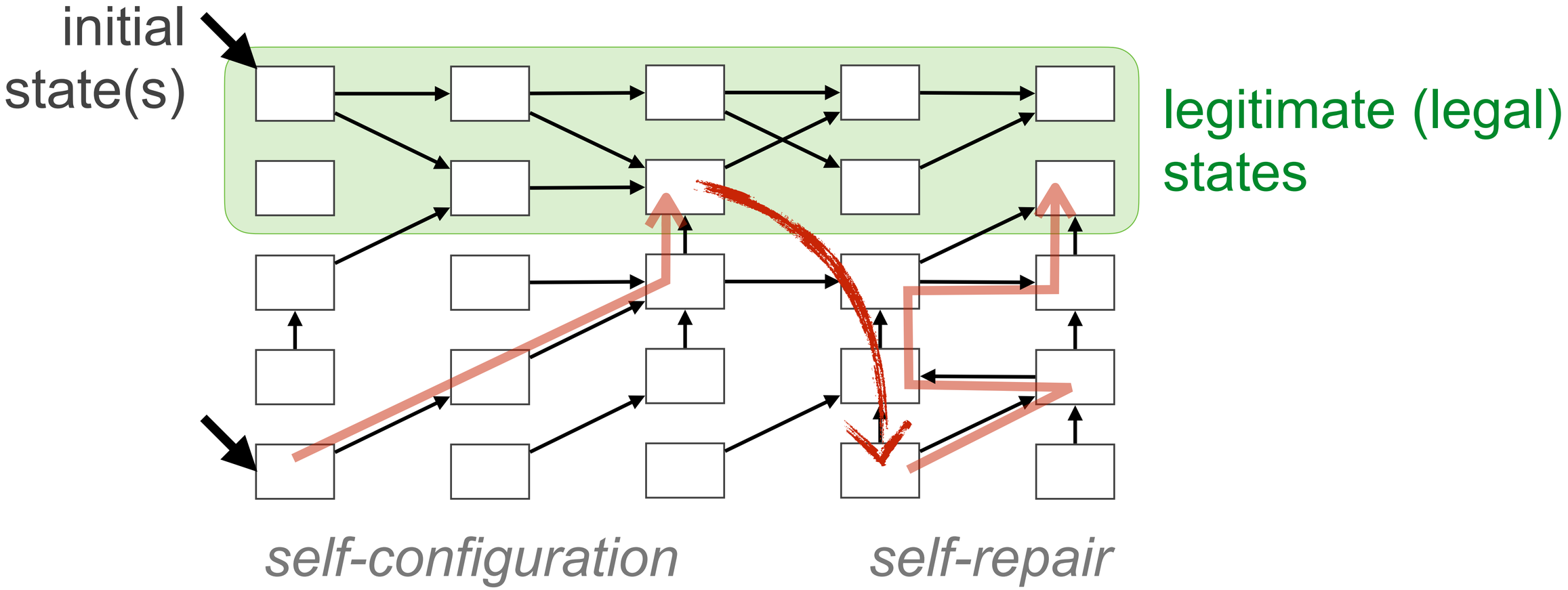
Classical Fault-Tolerance



Self-Stabilization



Self-Stabilization



system state

algorithm transitions

Self-Stabilization

Definition

- a system ***S*** is **self-stabilizing** with respect to predicate ***P***
  if it satisfies:

Closure

- ***P*** is closed under the execution of ***S***.
- *Intuitive*: always legal state —> legal state

Convergence

- starting from an arbitrary global state, ***S*** is guaranteed to reach a global state satisfying ***P*** within a finite number of state transitions.
- *Intuitive*: any state eventually —> legal state

Illustration: BFS tree

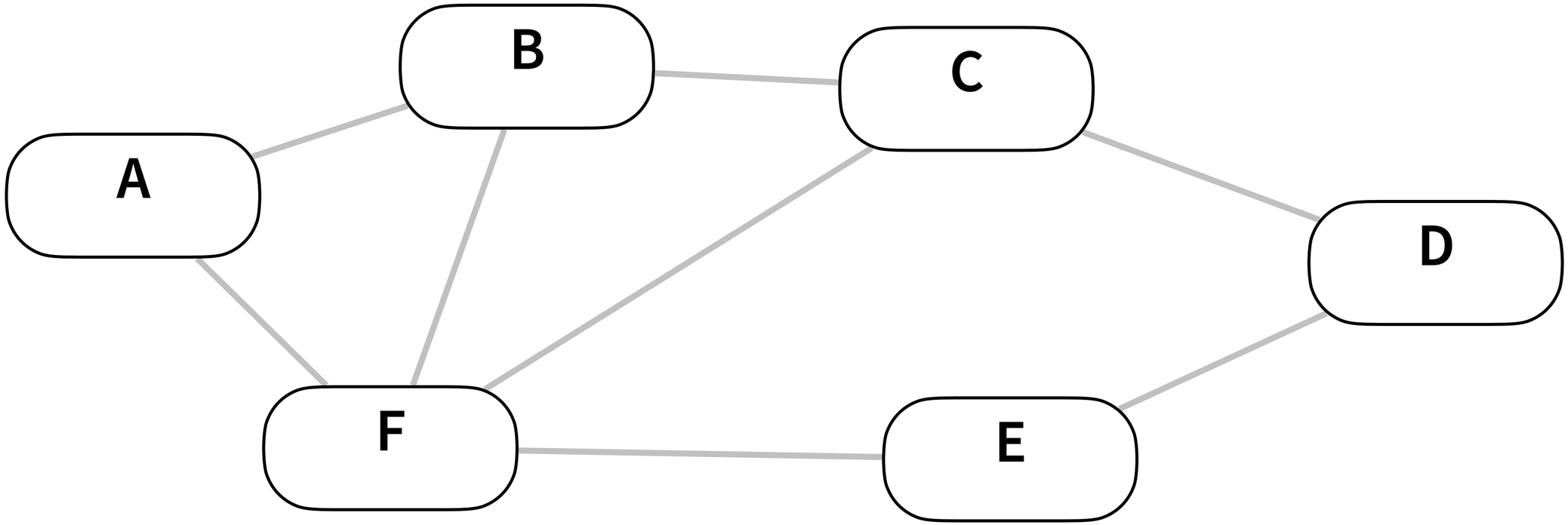


Illustration: BFS tree

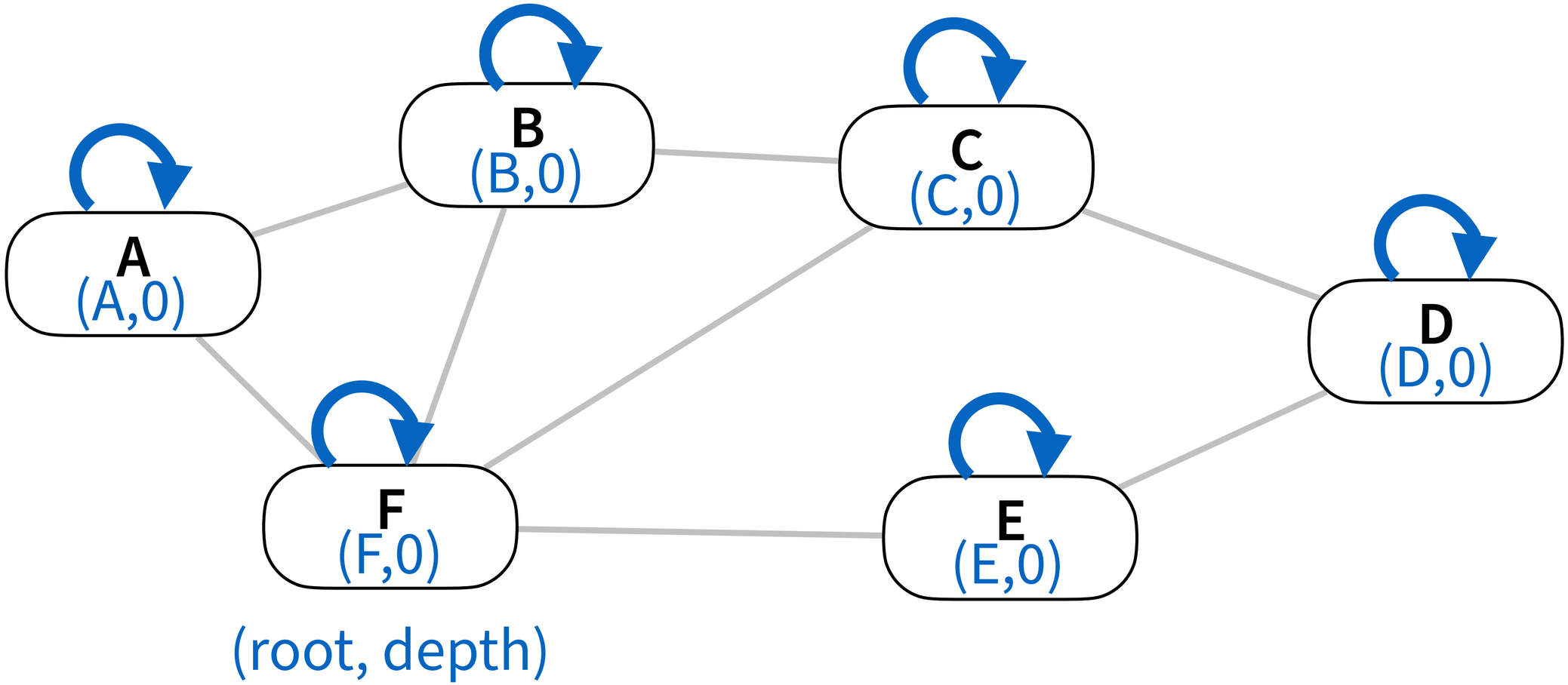


Illustration: BFS tree

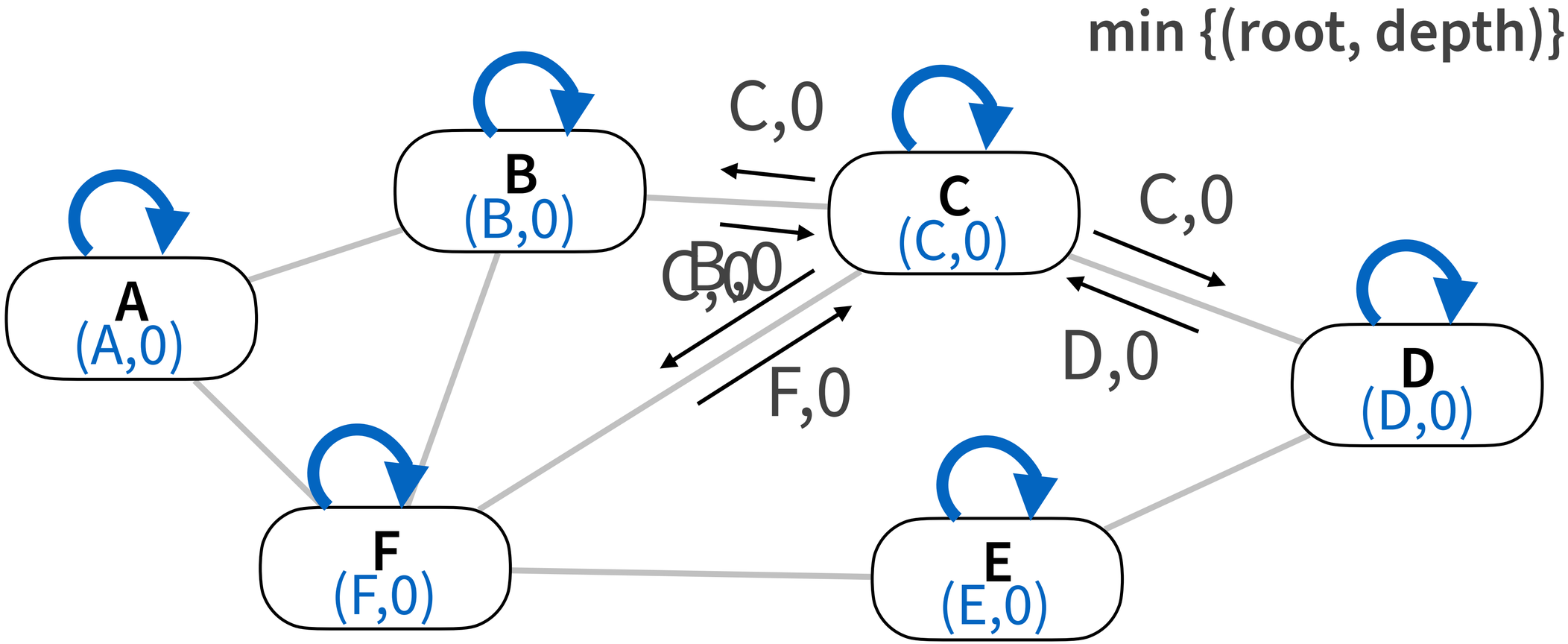


Illustration: BFS tree

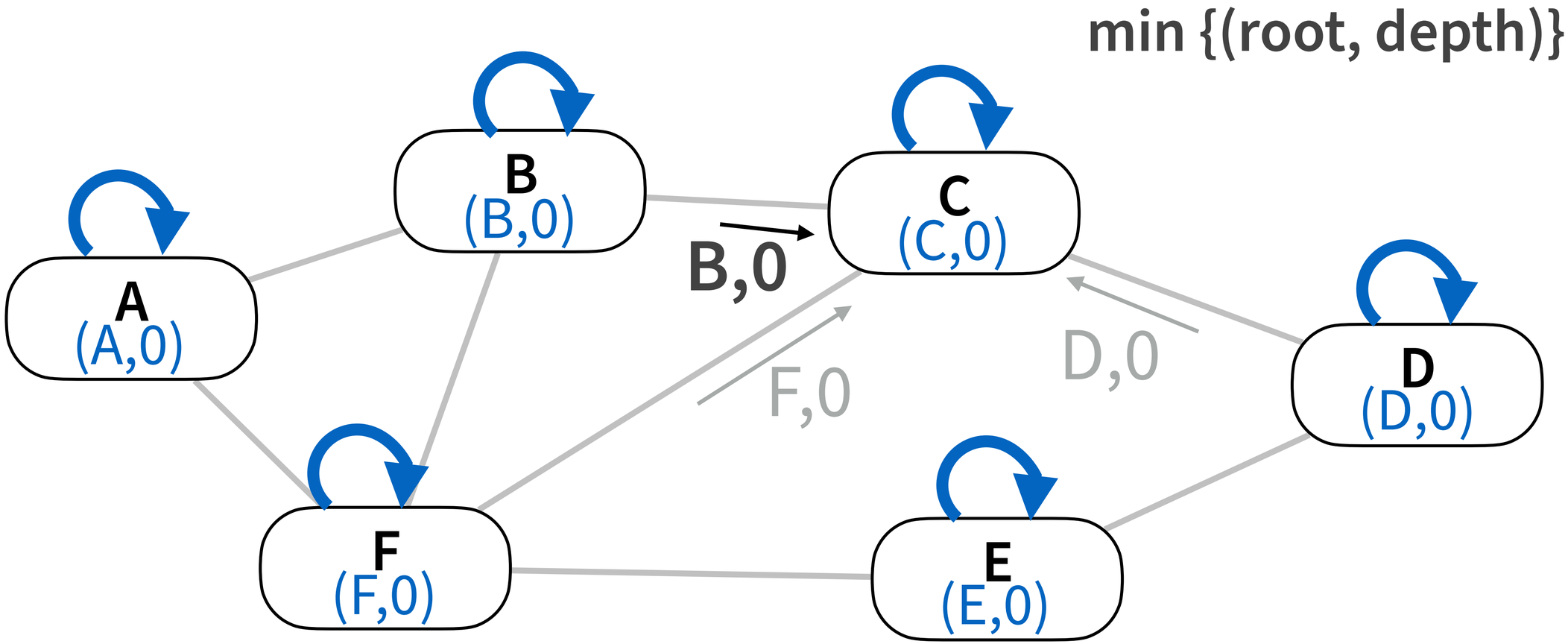


Illustration: BFS tree

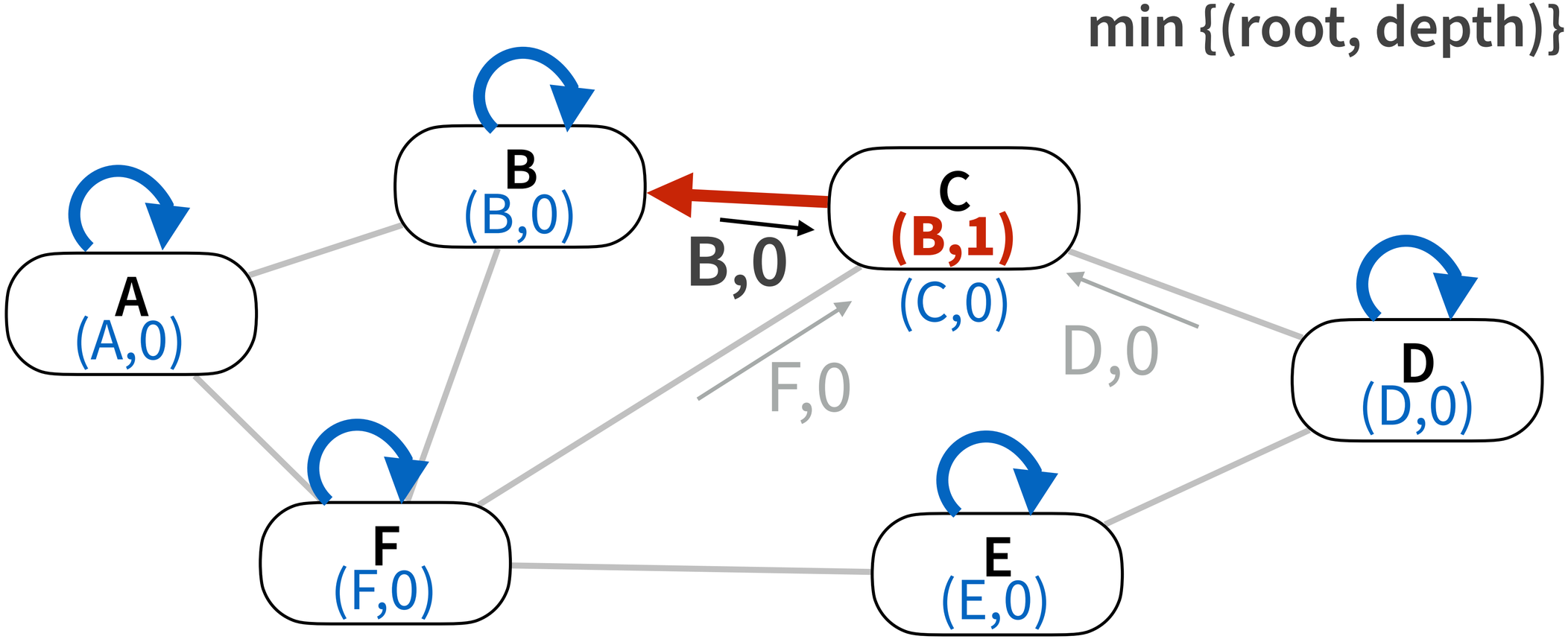


Illustration: BFS tree

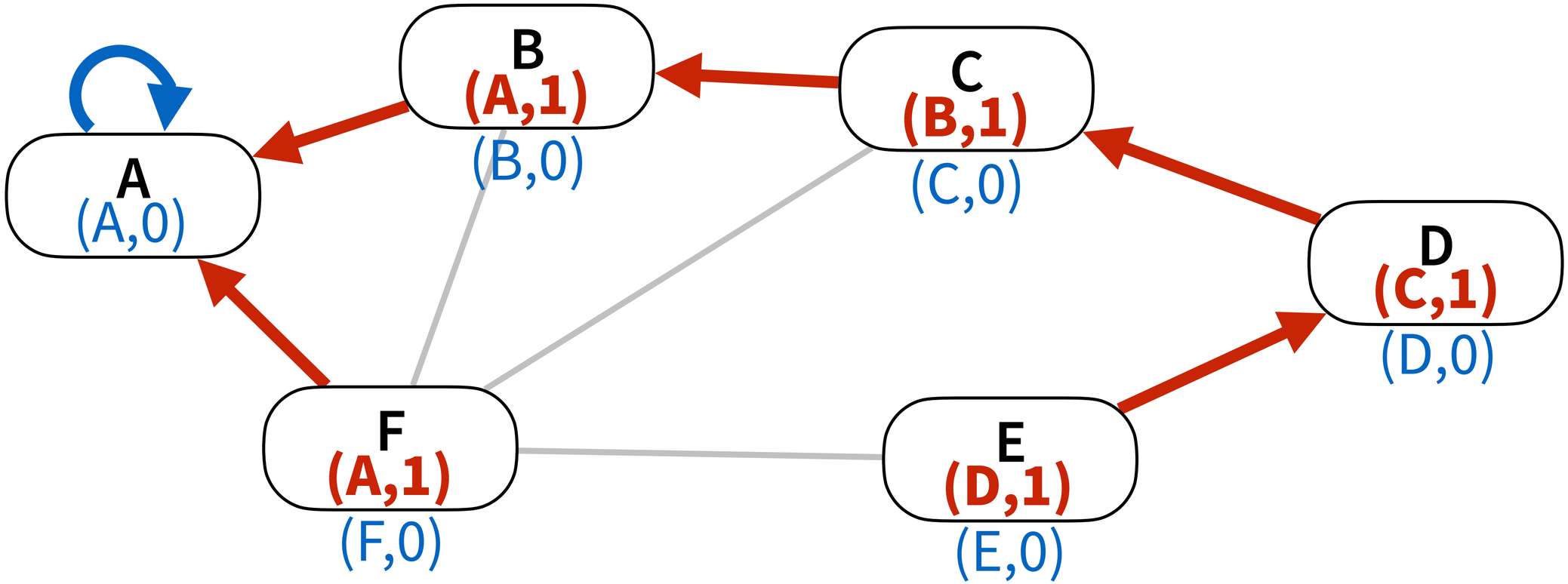


Illustration: BFS tree

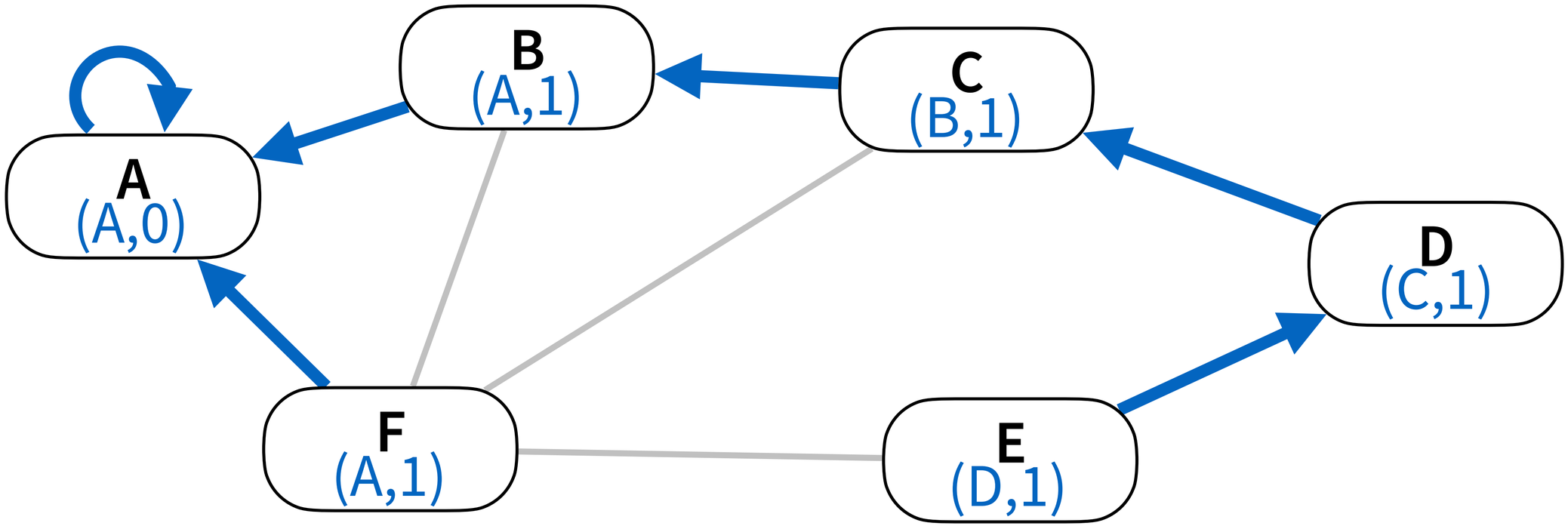


Illustration: BFS tree

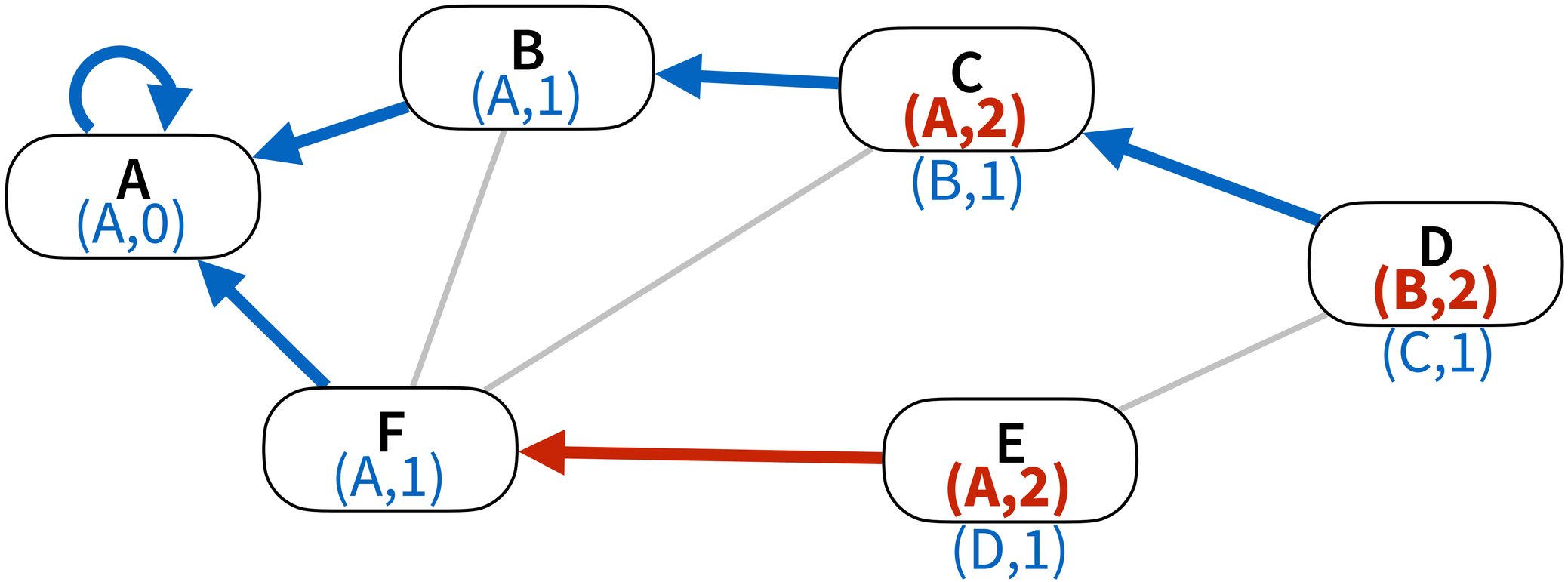


Illustration: BFS tree

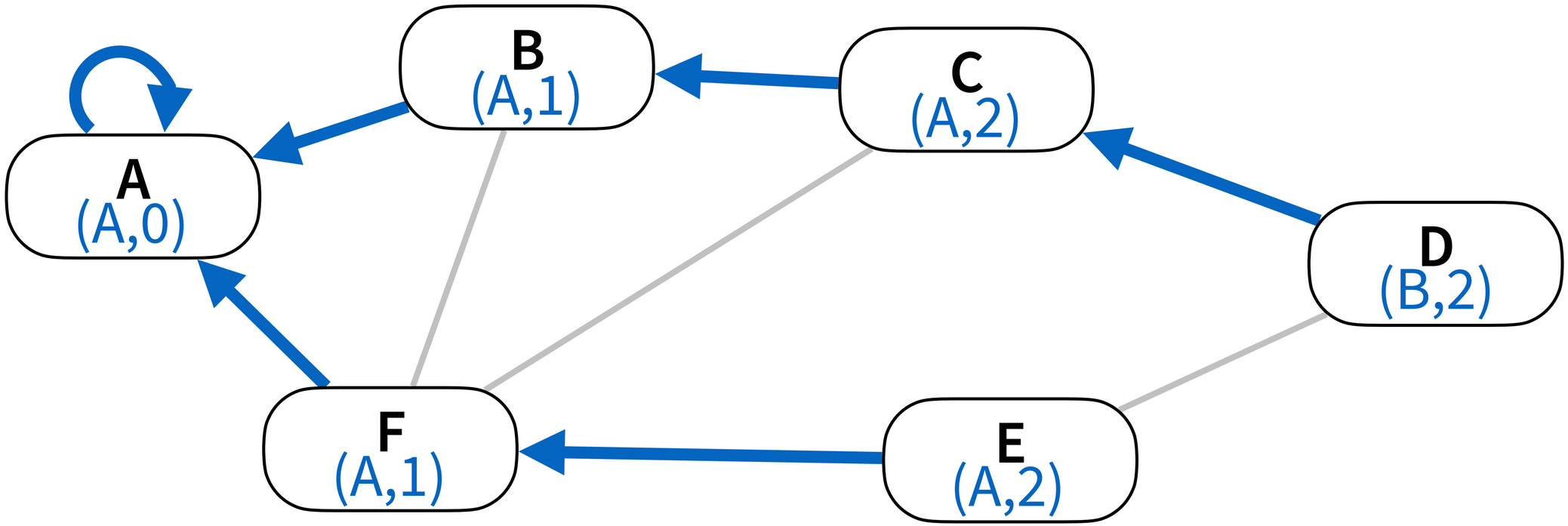


Illustration: BFS tree

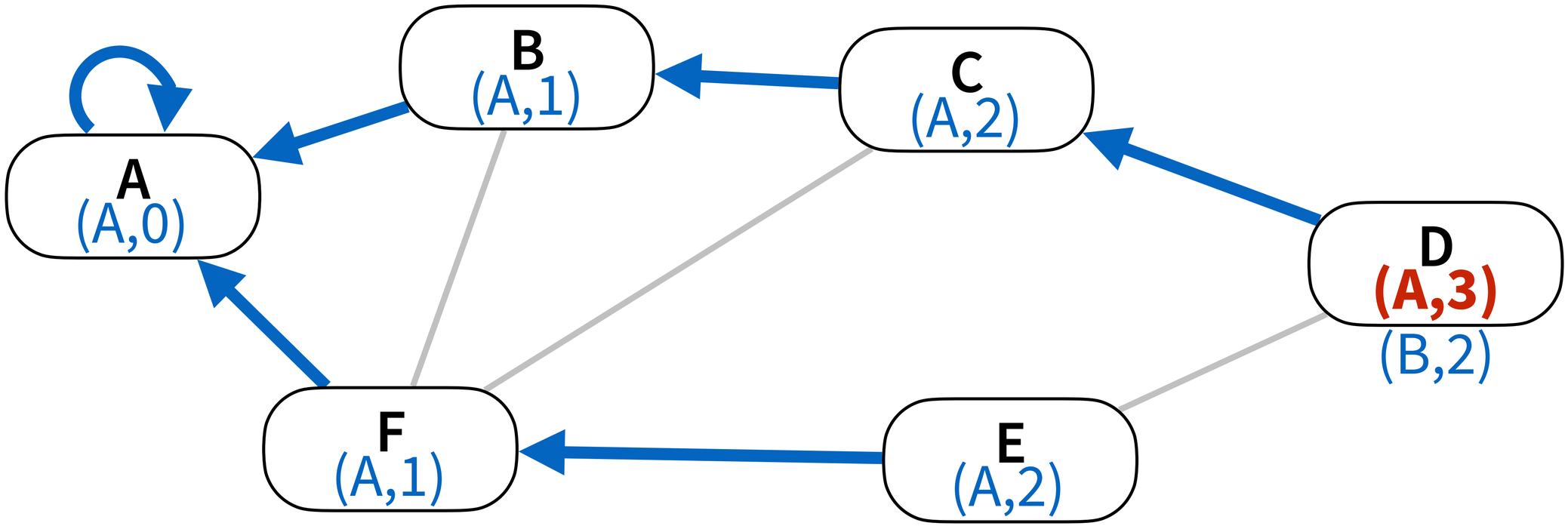


Illustration: BFS tree

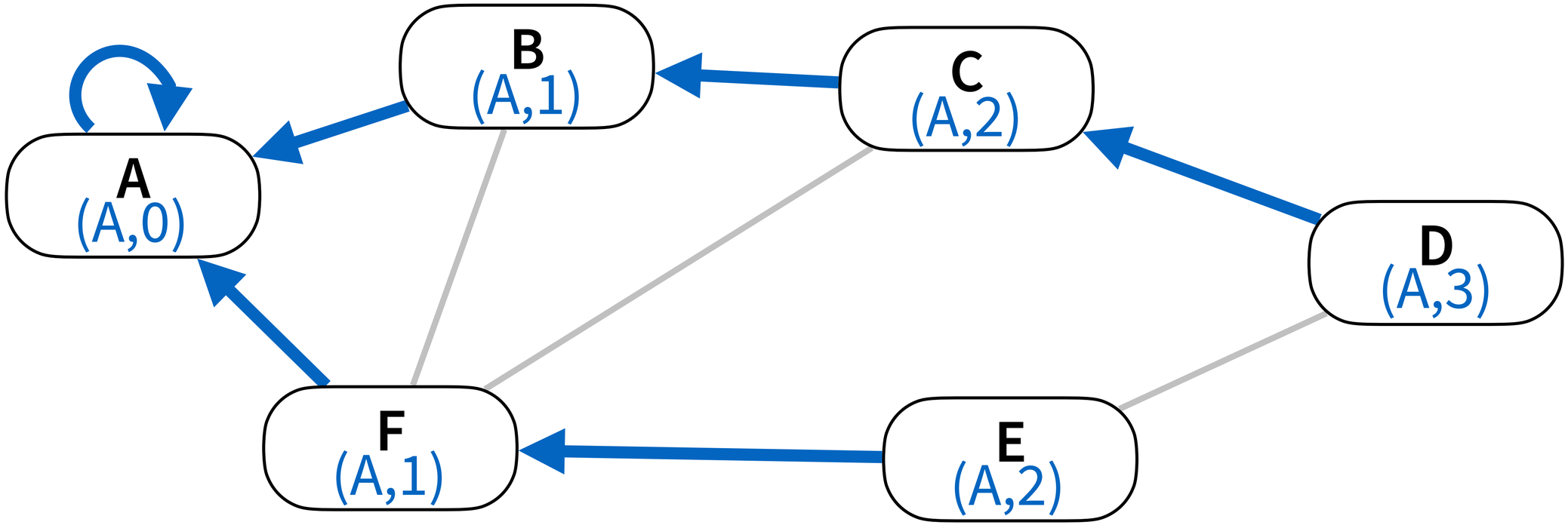


Illustration: BFS tree

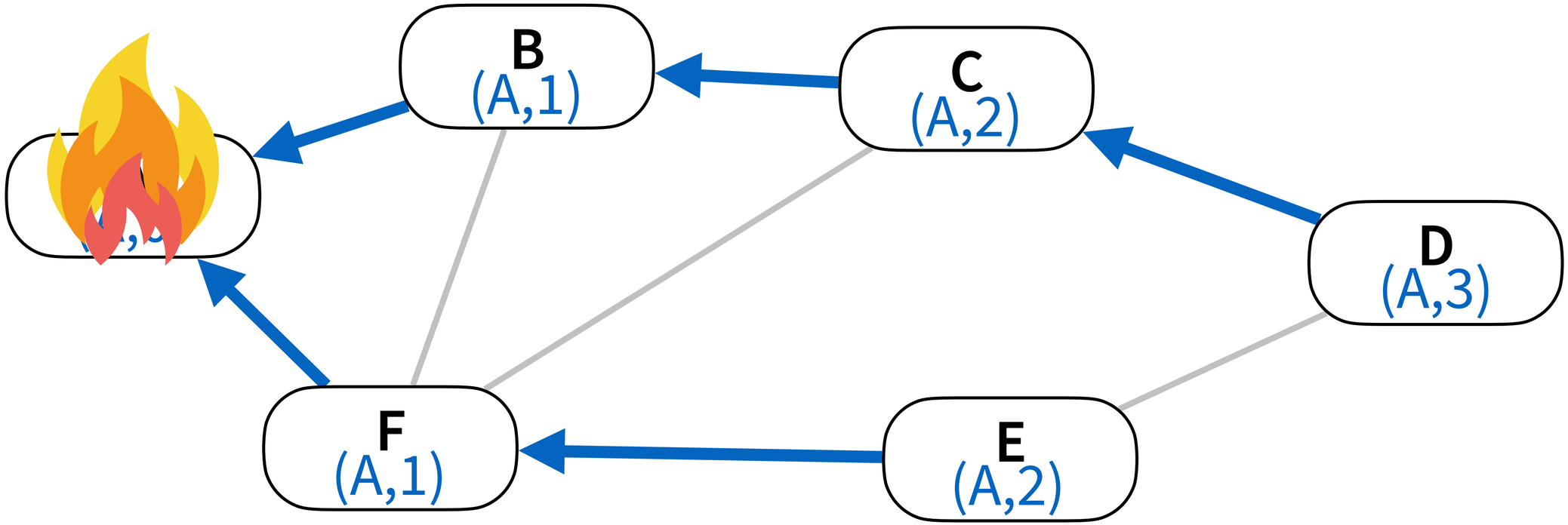


Illustration: BFS tree

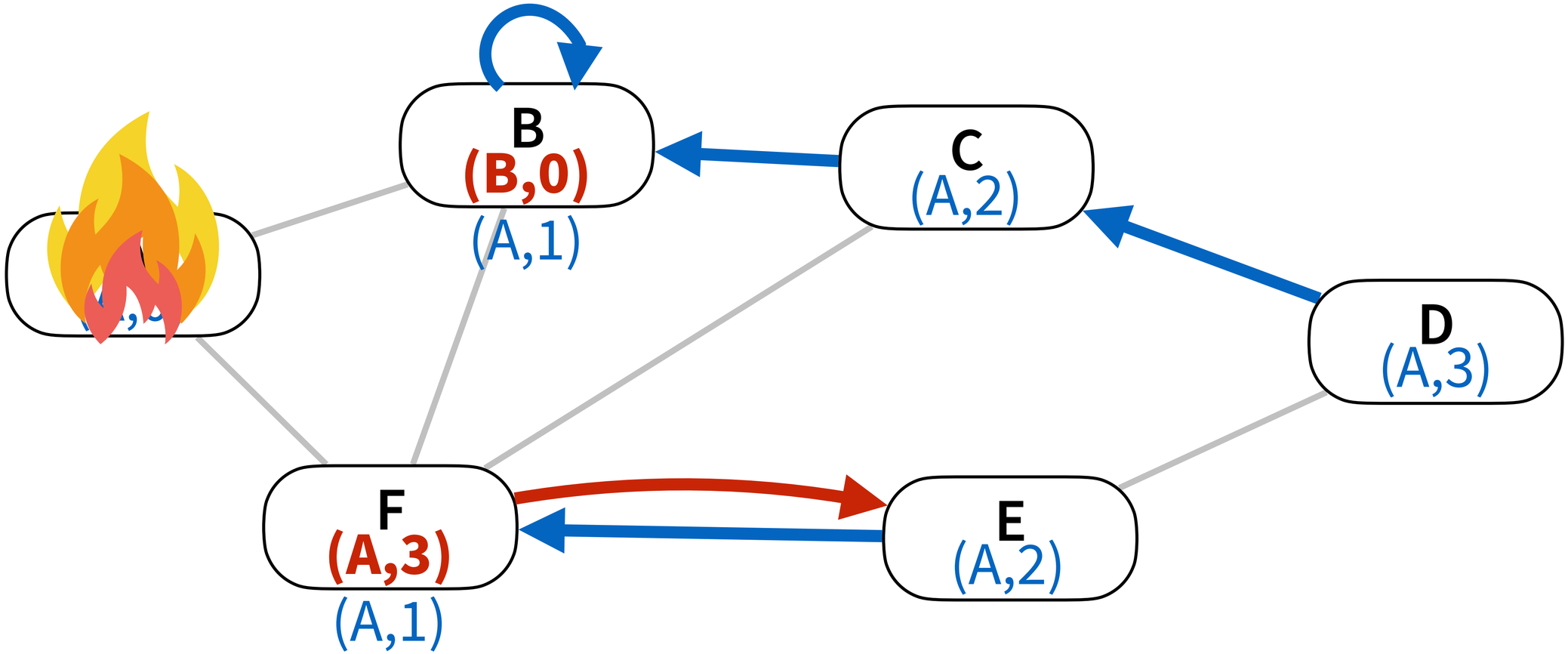


Illustration: BFS tree

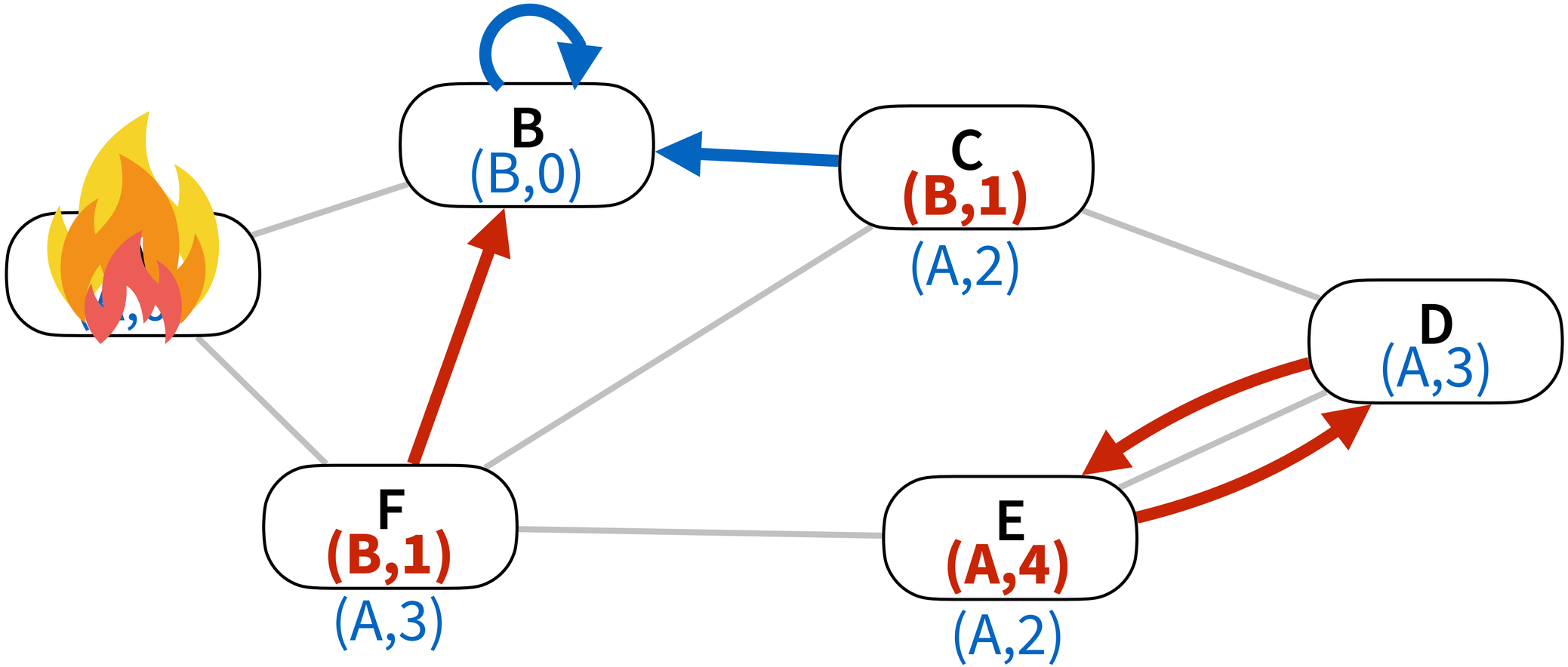


Illustration: BFS tree

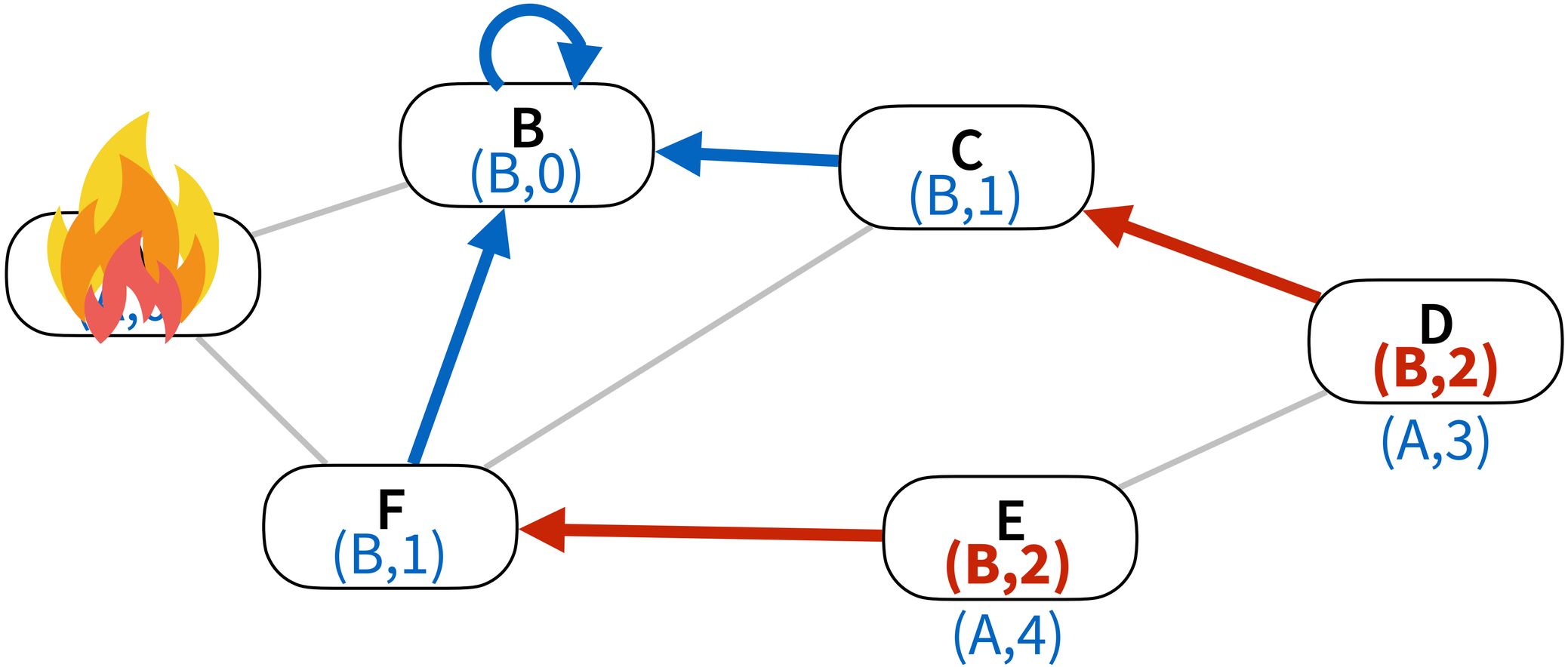
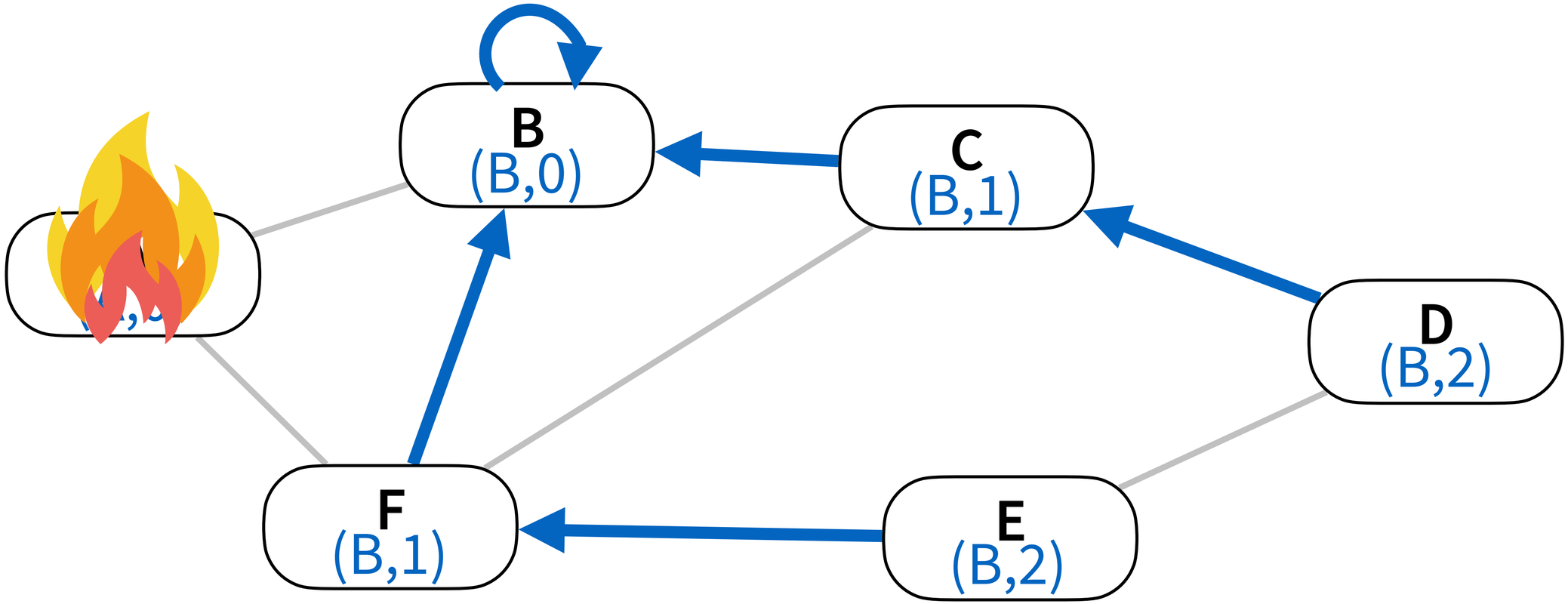


Illustration: BFS tree



Wrapup

Summary

- **Intrusion-Tolerance**
  - work in spite of traitors
  - covers for long detection time
  - complementary with other approaches (security in depth)
- **Distributed Agreement**
  - cornerstone
  - deal with potential traitors
- **Byzantine agreement**
  - robust but expensive
- **Blockchain**
  - weak but flexible
- **Self-stabilization**
  - attractive property
  - non-masking fault-tolerance