ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Replication & Consensus

(Slide credits: Lefteris Kokoris-Kogias & Enis Ceyhun Alp)

Outline

- Redundancy and Fault-Tolerance
- High Availability and Data Consistency
- Consensus
- Bitcoin & Blockchains

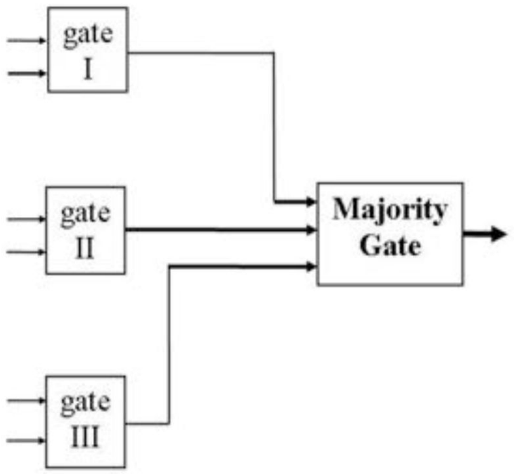
Outline

- **Redundancy and Fault-Tolerance**
- High Availability and Data Consistency
- Consensus
- Bitcoin & Blockchains

Redundancy

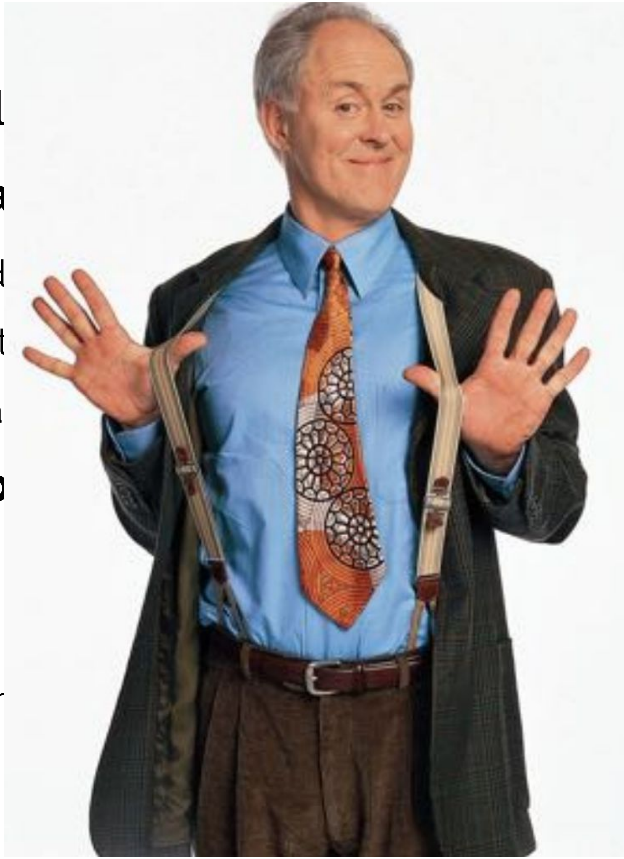
ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

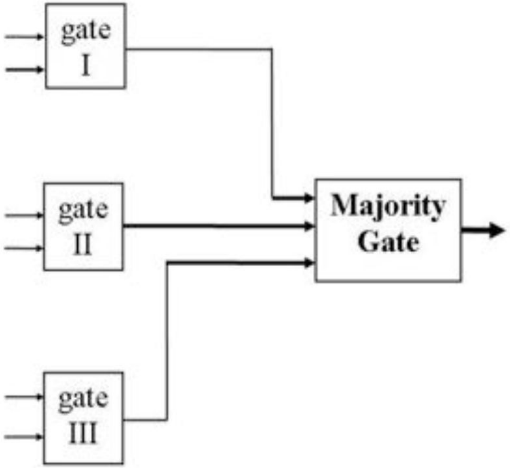
- Fundamental principle to build fault-tolerant systems
- Redundancy in **digital design**
  - Detect deviations and automatically restore correct behavior
  - Space-redundancy: state
  - Time-redundancy: transmission
- Redundancy in **computer systems**
  - Coding
  - Data replication
  - N-modular programming
  - Software replication



Redundancy

- Fundamental principl   ems
- Redundancy in **digita**
    - Detect deviations and   ior
    - Space-redundancy: st
    - Time-redundancy: tra
- Redundancy in **comp**
    - Coding
    - Data replication
    - N-modular programm
    - Software replication



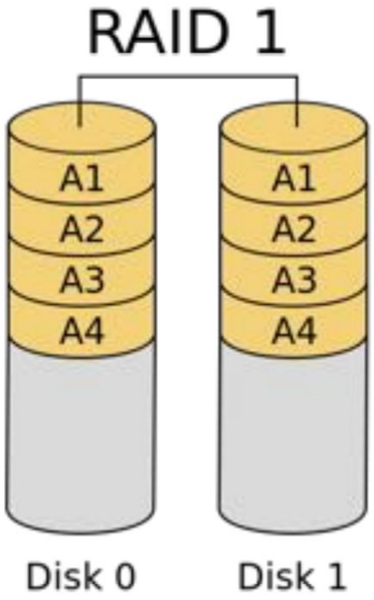


Redundancy Through Coding

- Incremental redundancy in memories:
  - DRAM ECC - correct single-bit errors, detect double-bit errors.
  - RAID5 -- symmetric parity encoding to recover from single-drive failures
  - RAID6 -- Galois-field encoding to recover from dual-drive failures.
- Incremental redundancy in communication
  - Forward-Error Correction (FEC) -- correct link errors on the link
  - Cyclic Redundancy Check (CRC) -- detect transmission errors on the link
- Incremental redundancy at the end-to-end layer
  - TCP checksum
  - SCSI -- Data Integrity Field (DIF)

Data Redundancy Through Replication

- RAID 1 – "mirroring"
  - 2 copies of each sector
  - Mechanism to detect disk failures
- Replication across systems
  - Copies in different location
  - For availability, disaster recovery, or content distribution
  - Strongly or weakly consistent variants
- Example – cloud storage (HDFS, Amazon S3)
  - 3 independent copies



Fault Tolerance

- Denial is not a strategy – things will fail
  - Your code
  - Your computer
  - Somebody else's code
  - Some part of the environment

Definitions

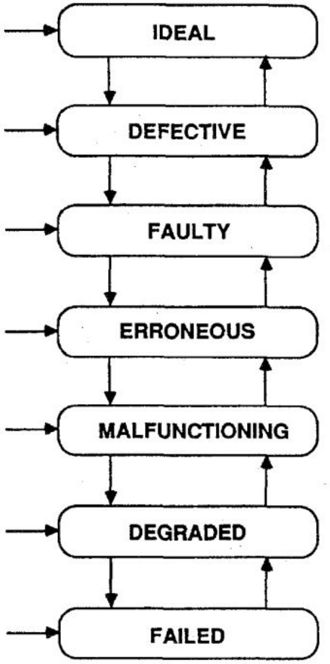
Fault → underlying defect, e.g. software (bug), hardware (fried component), operation (user error), environment (power grid)

- Can be active (generates errors) or latent

Failure → module not producing the desired result, e.g. an error

- Occurs when a fault is not detected and masked by the module

Fault tolerance → building reliable systems out of unreliable components

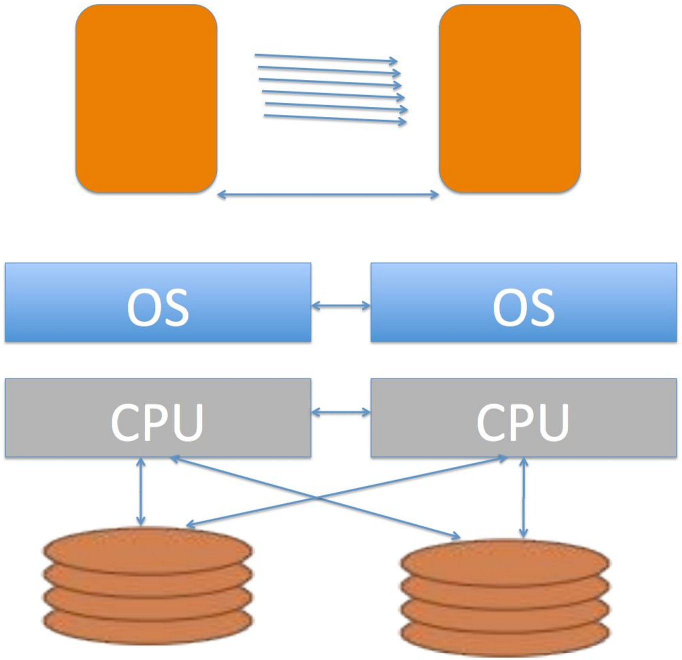


Tolerating software faults

- Applying NMR to software → N-version programming
  - Example: DNS root servers run on different systems with different implementations
  - Flight-control systems (Swiss Boeing 777 -- N=3)
- Systematic approaches to fault tolerance in systems
  - Respond to active faults (within a system) → containment + repair
  - Examples
    - Process pairs
    - High-availability clusters
    - Consensus algorithms

Tandem NONSTOP

- Redundant hardware components
- Process pairs
  - Each process has a backup
  - API to communicate state changes using messages
  - Process heartbeat to detect failures at all levels
- Fast detection (fail-fast)
- Fast recovery of transient software faults (process pairs)



Outline

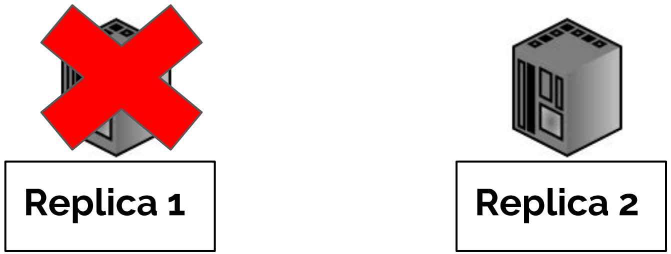
- Redundancy and Fault-Tolerance
- **High Availability and Data Consistency**
- Consensus
- Bitcoin & Blockchains
- Smart Contracts

Replication Technique

- Distributed systems replicate data across multiple servers

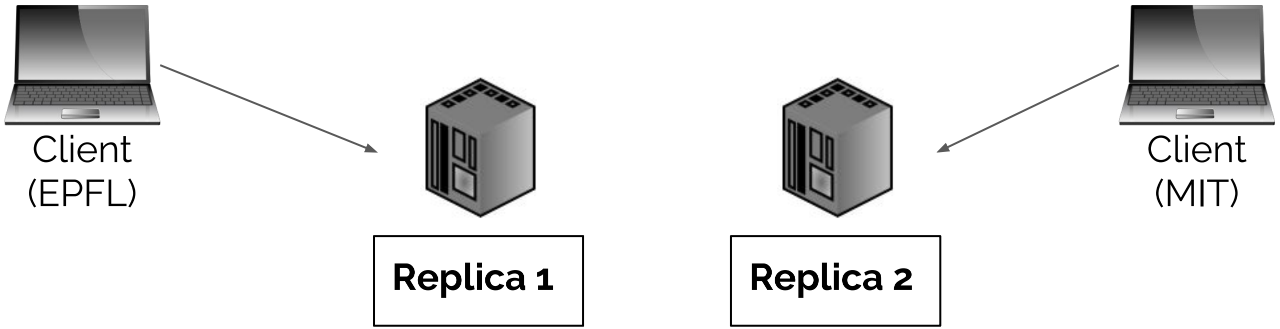
Replication Technique

- Distributed systems replicate data across multiple servers
  - Replication provides fault-tolerance if servers fail



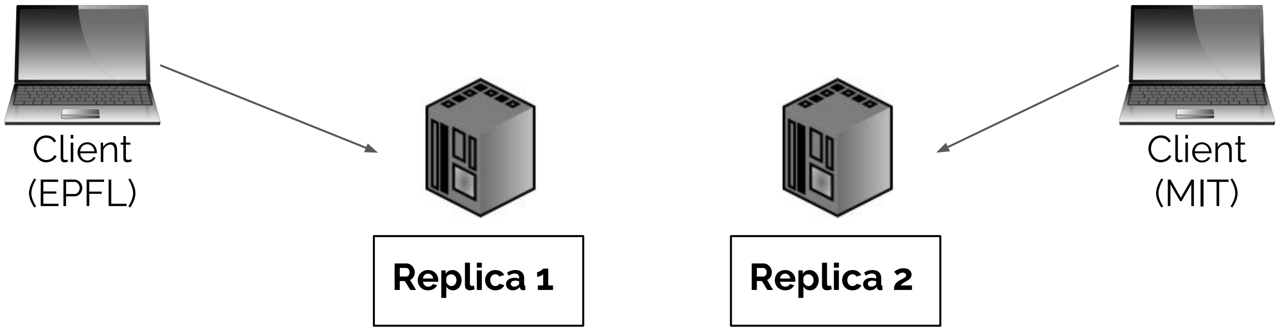
Replication Technique

- Distributed systems replicate data across multiple servers
  - Replication provides fault-tolerance if servers fail
  - Allowing clients to access different servers potentially increasing scalability (max throughput)

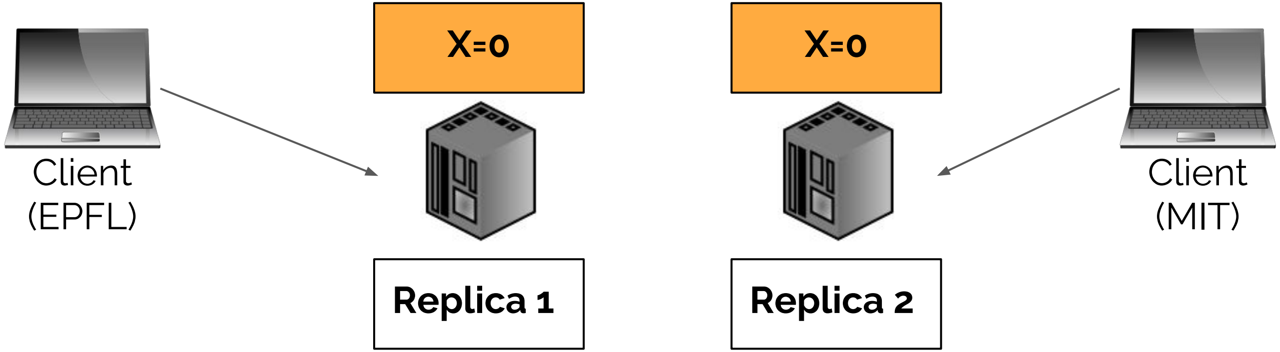


Replication Technique

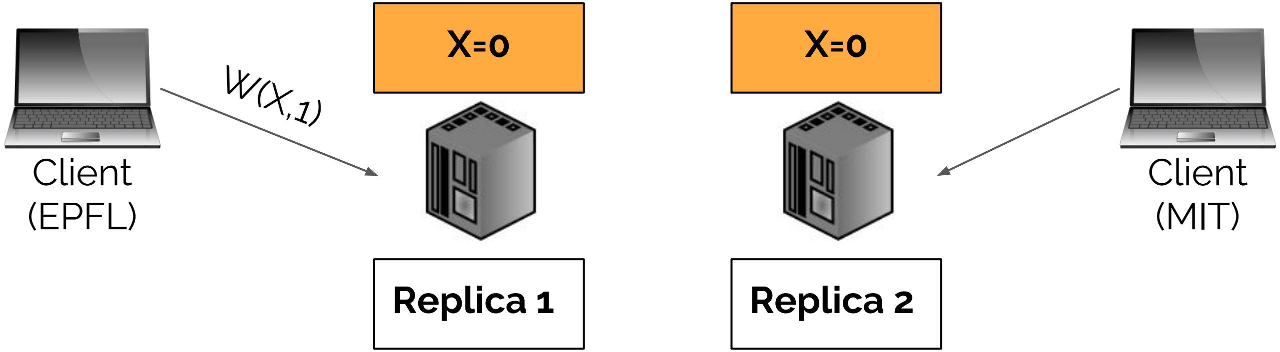
- Distributed systems replicate data across multiple servers
  - Replication provides fault-tolerance if servers fail
  - Allowing clients to access different servers potentially increasing scalability (max throughput)
  - **What is the problem?**



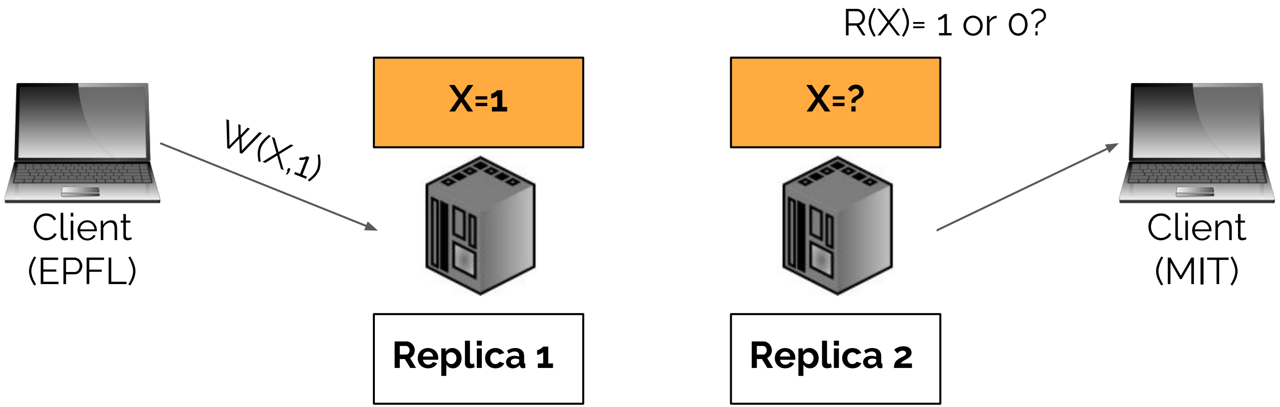
Consistency Problem



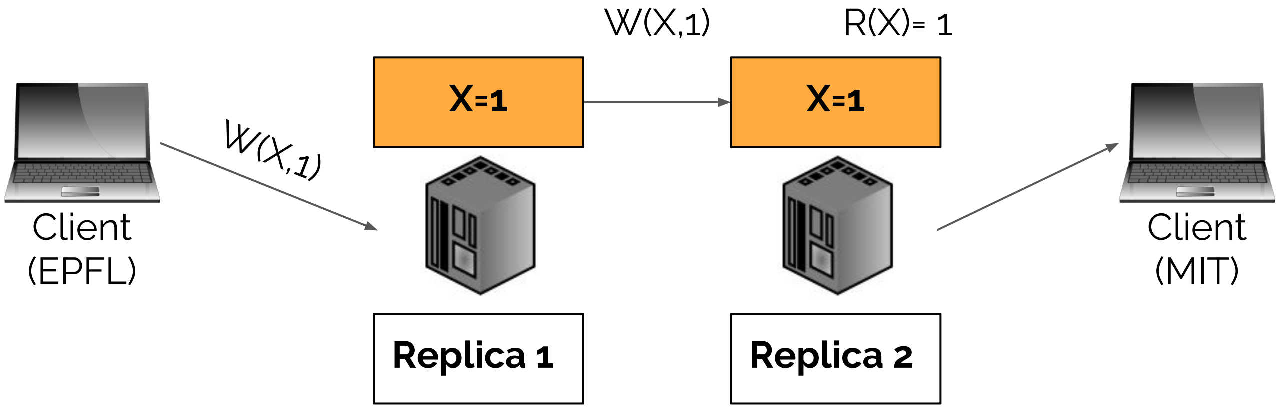
Consistency Problem



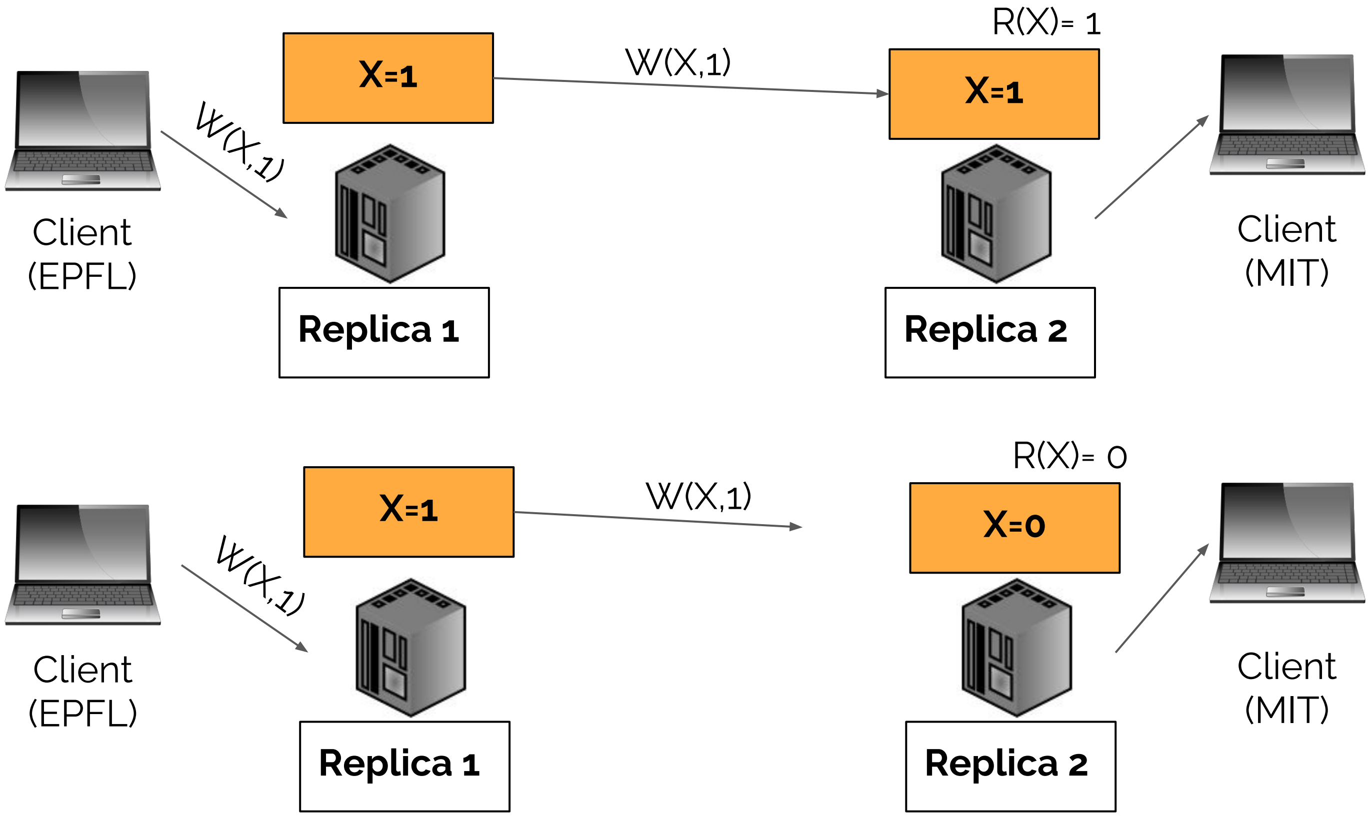
Consistency Problem



Consistency Problem



Consistency Problem



Disclaimer for Databases

- Atomicity
- **Consistency → Not that kind of consistency!!**
- Integrity
- Durability

Outline

- Redundancy and Fault-Tolerance
- **High Availability and Data Consistency**
- Consensus
- Bitcoin & Blockchains

Consistency Models

- A consistency model specifies a contract between **programmer** and **system**, wherein the system guarantees that if the programmer follows the rules, data will be consistent

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

Consistency Models

- A consistency model specifies a contract between **programmer** and **system**, wherein the system guarantees that if the programmer follows the rules, data will be consistent
- If a system supports the stronger consistency model, then the weaker consistency model is automatically supported

Consistency Models

- A consistency model specifies a contract between **programmer** and **system**, wherein the system guarantees that if the programmer follows the rules, data will be consistent
- If a system supports the stronger consistency model, then the weaker consistency model is automatically supported
- But stronger consistency models sacrifice more availability and fault tolerance

Many Consistency Models

- Strict Consistency
- Linearizability
- Sequential Consistency
- Causal Consistency
- Eventual Consistency

Weaker consistency models

These models describe when and how different nodes in a distributed system view the order of operations

Many Consistency Models

- Strict Consistency
- Lineariz
- Sequent
- Causal C
- Eventua

Why we have so many consistency models?

Many Consistency Models

- Strict Consistency
- Lineariz
- Sequent
- Causal C
- Eventua

Why we have so many consistency models?

Different applications → different trade-offs between consistency/availability/fault-tolerance

Strong Consistency

- Strict Consistency
- Linearizability
- Sequential Consistency
- Causal Consistency
- Eventual Consistency

Weaker consistency models

Strong Consistency

- Strict Consistency
- **Linearizability**
- Sequential Consistency
- Causal Consistency
- Eventual Consistency

Weaker consistency models

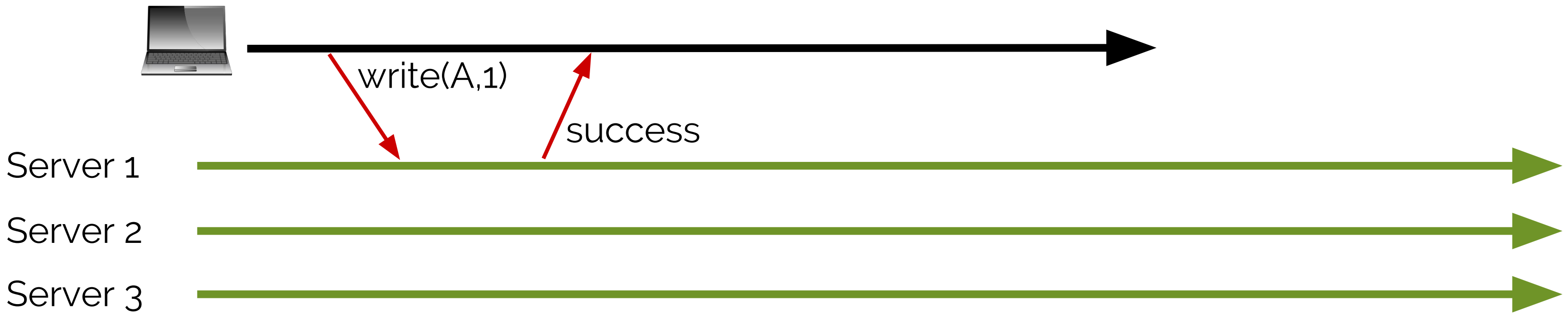
Linearizability

- Provide behavior of a single copy of object
  - Read should return the most recent write
  - Subsequent reads should return same value, until next write

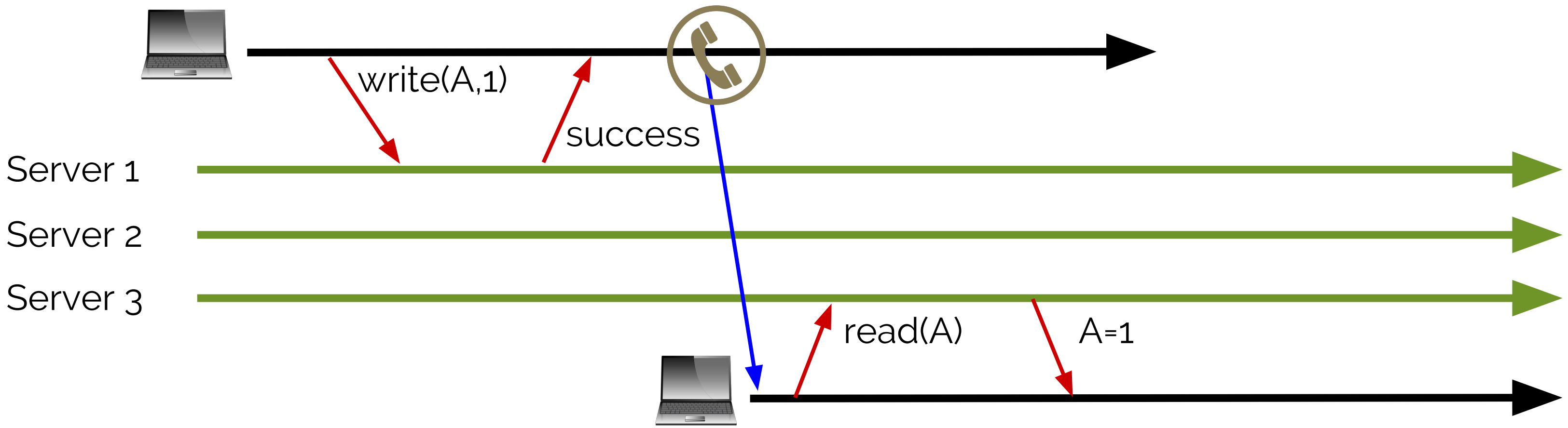
Linearizability

- Provide behavior of a single copy of object:
  - Read should return the most recent write
  - Subsequent reads should return same value, until next write
- Telephone intuition:
  - Bob updates Facebook post
  - Bob calls Alice on phone: “Check my Facebook post!”
  - Alice read’s Bob’s wall, sees his post

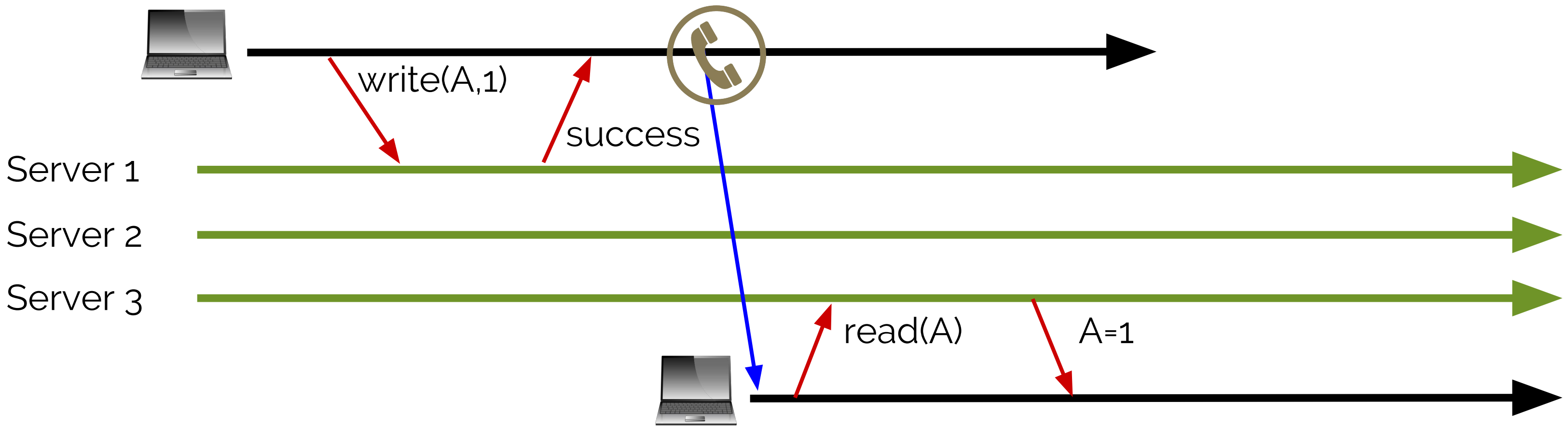
Linearizability



Linearizability



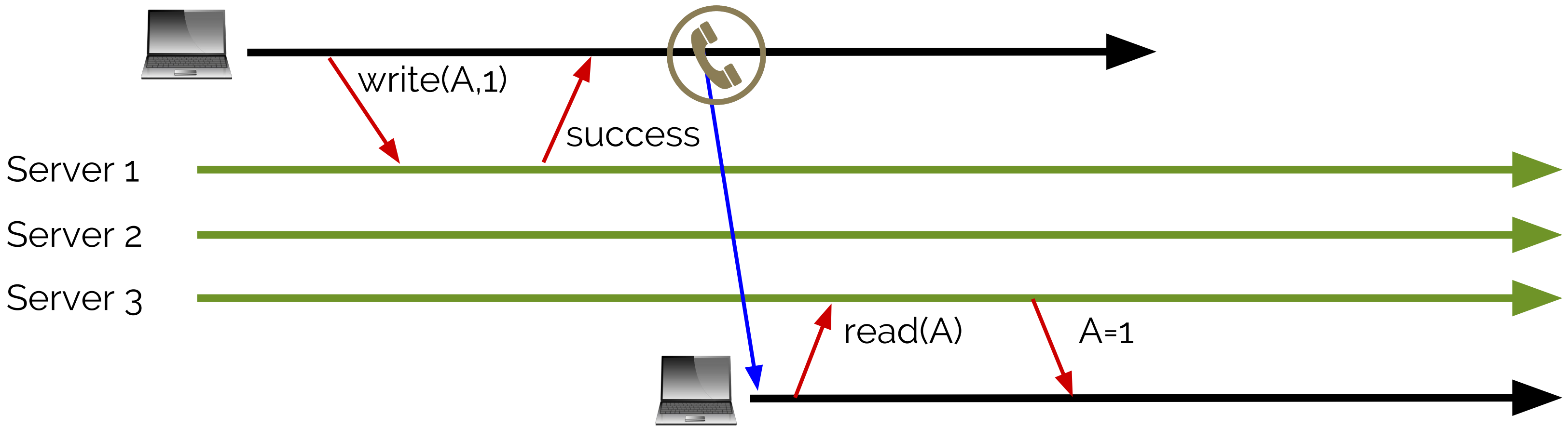
Linearizability



How to achieve this?

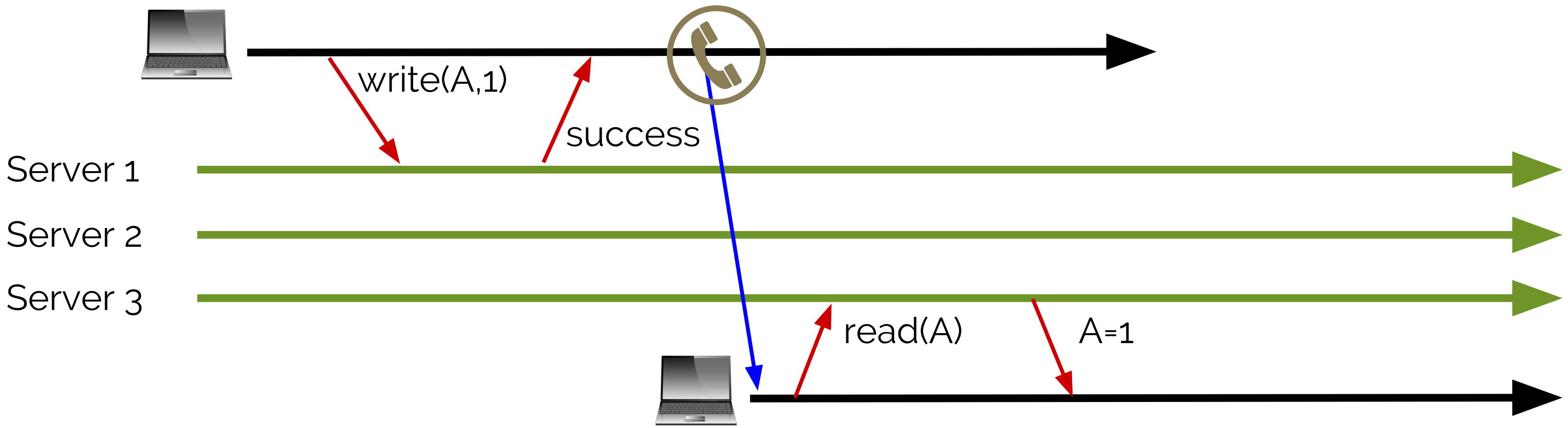
Server 3 did not get the write

Linearizability



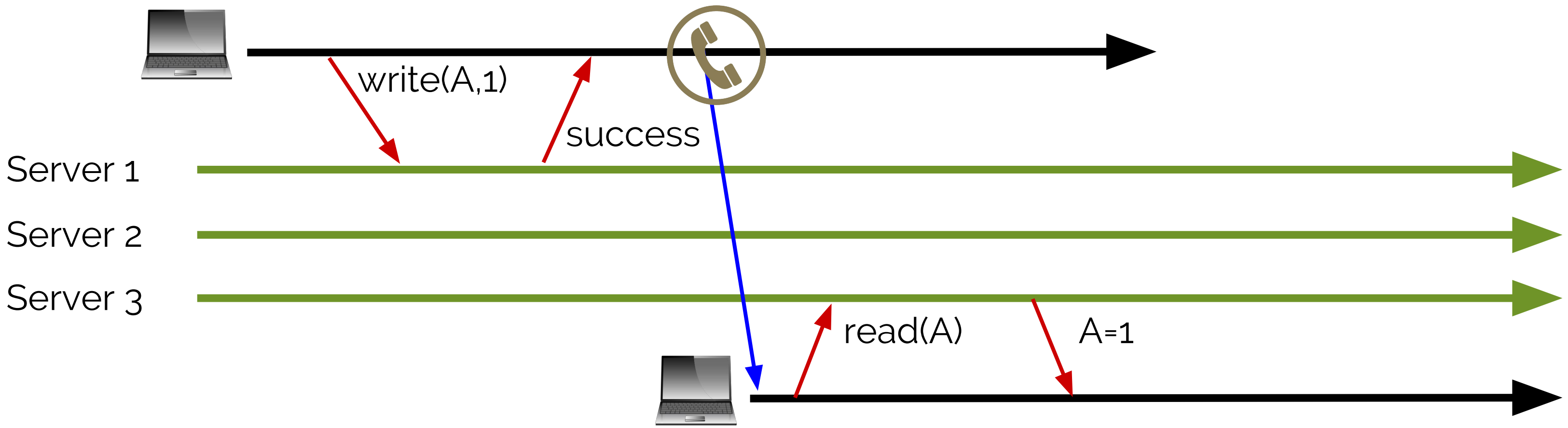
Idea: Delay responding to writes/ops until committed

Linearizability? This is buggy!



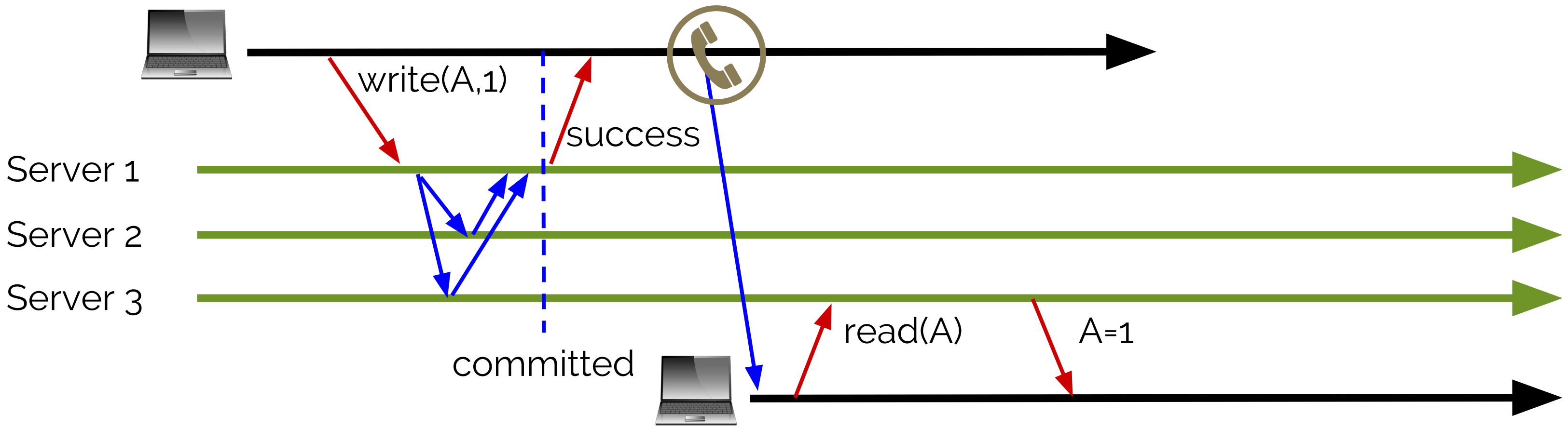
Idea: Delay responding to writes/ops until committed

Linearizability? This is buggy!



- How much delay is “enough”? Who writes on Server 3?
- Not sufficient to return value of Server 3 → It does not know precisely when op is “globally” committed
- Need global ordering between the write and the read operation

Linearizability!



Order all operations via (1) leader and (2) agreement

Linearizability

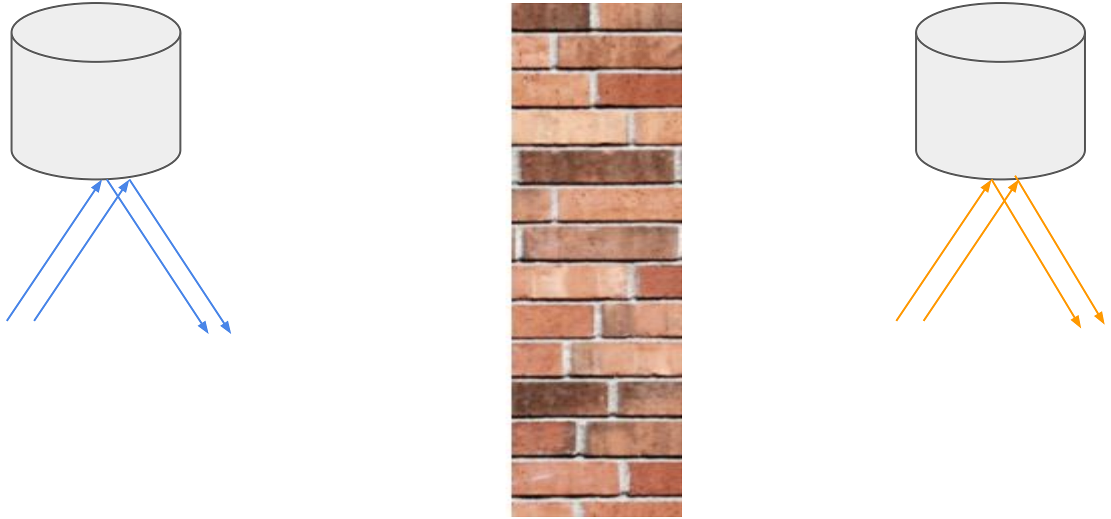
- Linearizability:
  - All servers execute all ops in some identical sequential order
  - Global ordering preserves each client's own local ordering
- Once write completes, **all later reads should return value of that write or value of later write**
- Once read returns particular value, **all later reads should return that value or value of later write**

High Availability

High Availability

System guarantees a response, even during network partitions (async network)

[Gilbert and Lynch, ACM SIGACT News 2002]



Network partitions

"Network partitions should be rare but net gear continues to cause more issues than it should." --James Hamilton, Amazon Web Services

[perspectives.mvdirona.com, 2010]

MSFT LAN: avg. 40.8 failures/day (95^{th} %ile: 136) 5 min median time to repair (up to 1 week)

[SIGCOMM 2011]

HP LAN: 67.1% of support tickets are due to network median incident duration 114-188 min

[HP Labs 2012]

Weak Consistency

- Strict Consistency
- Linearizability
- Sequential Consistency
- **Causal Consistency**
- Eventual Consistency

Weaker consistency models

Causal Consistency

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

- Causal consistency is one of weak consistency models
  - Causally related writes must be seen by all processes in the same order
  - Concurrent writes may be seen in different orders on different machines

Causal Consistency

- Have you seen causal consistency?
- Have you implemented causal consistency?

Weak Consistency

- Strict Consistency
- Linearizability
- Sequential Consistency
- Causal Consistency
- **Eventual Consistency**

Weaker consistency models

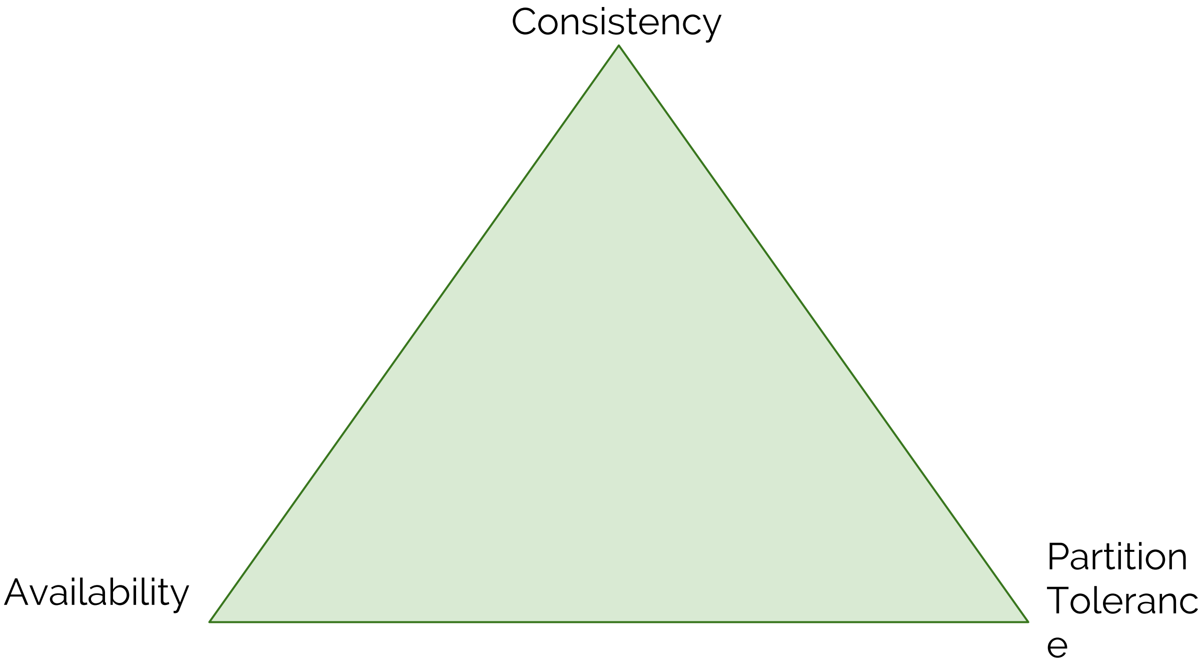
Eventual Consistency

- Eventual consistency
  - Achieve high availability
  - If no new updates are made to a given data item, eventually all accesses to the data will return the last updated value
- Eventual consistency is commonly used
  - Git repo, iPhone sync
  - Dropbox
  - Amazon Dynamo

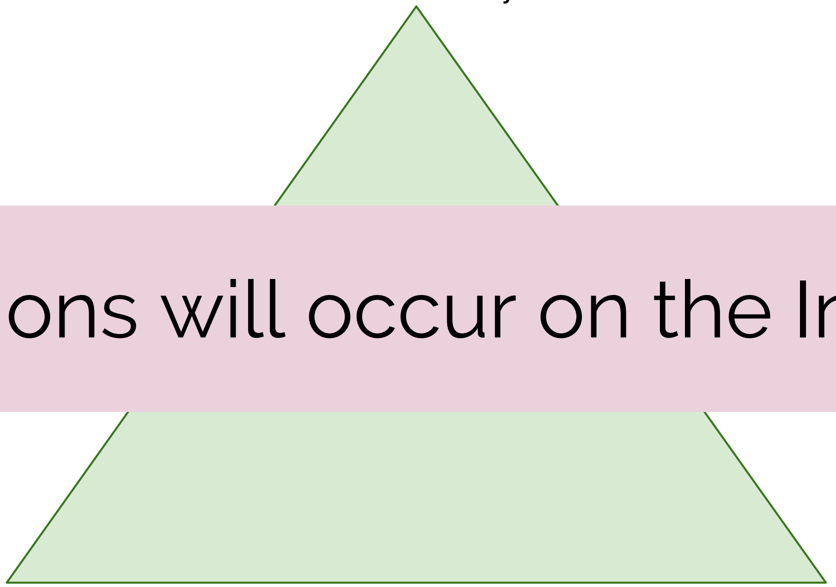
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

The **CAP** Theorem

The CAP Theorem



The CAP Theorem



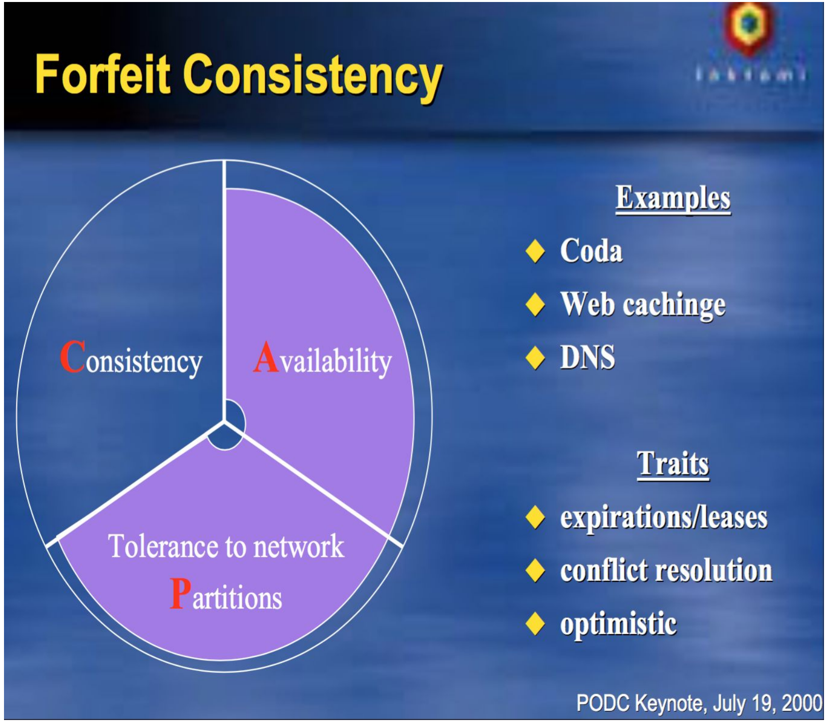
Partitions will occur on the Internet

Disclaimer

- CAP is not as absolute as many claim
    - "Highly Available Transactions: Virtues and Limitations", P.Bailis et al. VLDB 2014
    - "CAP Twelve Years Later: How the "Rules" Have Changed", E.Brewer, Computer 45.2 (2012)

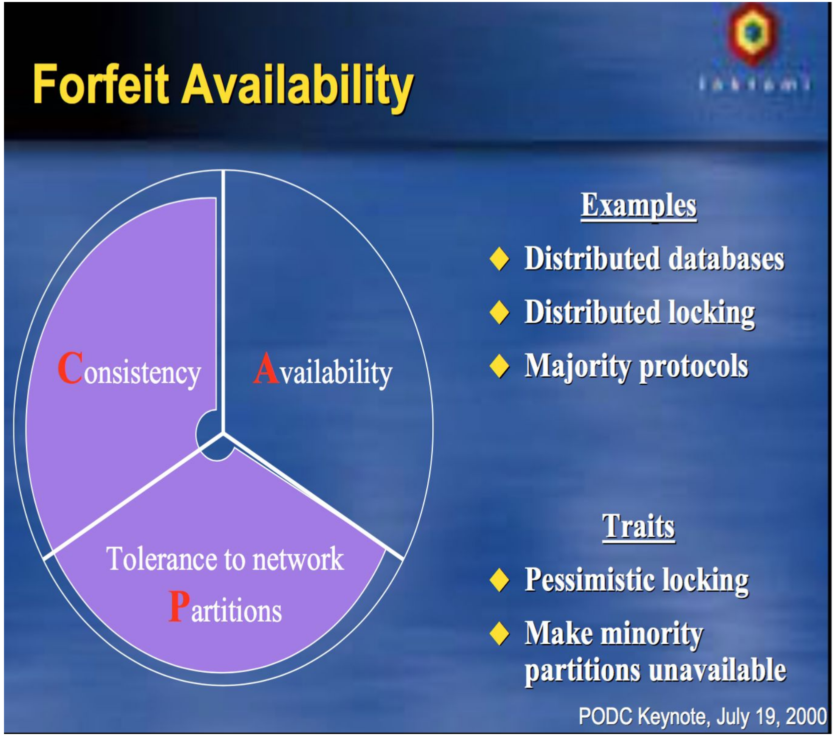
The AP Choice

- Strong consistency is not possible
  - The system can reply with stale data
- Many applications do not care
  - DNS
  - Web caching
  - Most applications (e.g., Facebook, Dropbox)
- Benefits of weak consistency
  - Highly-available systems
  - Low latency
  - No coordination



The CP Choice

- Strong Consistency
  - Safety first
  - System halts on partitions
- Needs coordination
  - Consensus protocols
- Benefits
  - Writes are atomic
  - Any data read are the freshest possible



Outline

- Redundancy and Fault-Tolerance
- High Availability and Data Consistency
- **Consensus**
- Bitcoin & Blockchains

Consensus

- In the consensus problem, processes propose values and have to agree on one of these values
- Properties
  - Validity: Any value decided is a value proposed
  - Agreement: No two correct processes decide differently
  - Termination: Every correct process eventually decides
  - Integrity: No process decides twice

Round Synchronous

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

- The processes go through rounds incrementally (1 to n)
  - In each round, the process with the id corresponding to that round is the leader of the round
- The leader of a round decides its current proposal and broadcasts it to all
- A process that is not leader in a round waits:
  - (a) to deliver the proposal of the leader in that round to adopt it **OR**
  - (b) to suspect the leader

Uniform Consensus Algorithm

- The processes go through rounds incrementally (1 to n)
    - In each round i, process p_i sends its current ***proposal*** to all
- A process adopts any current ***proposal*** it receives
- Processes decide on their current ***proposal*** values at the end of round n

Asynchronous?

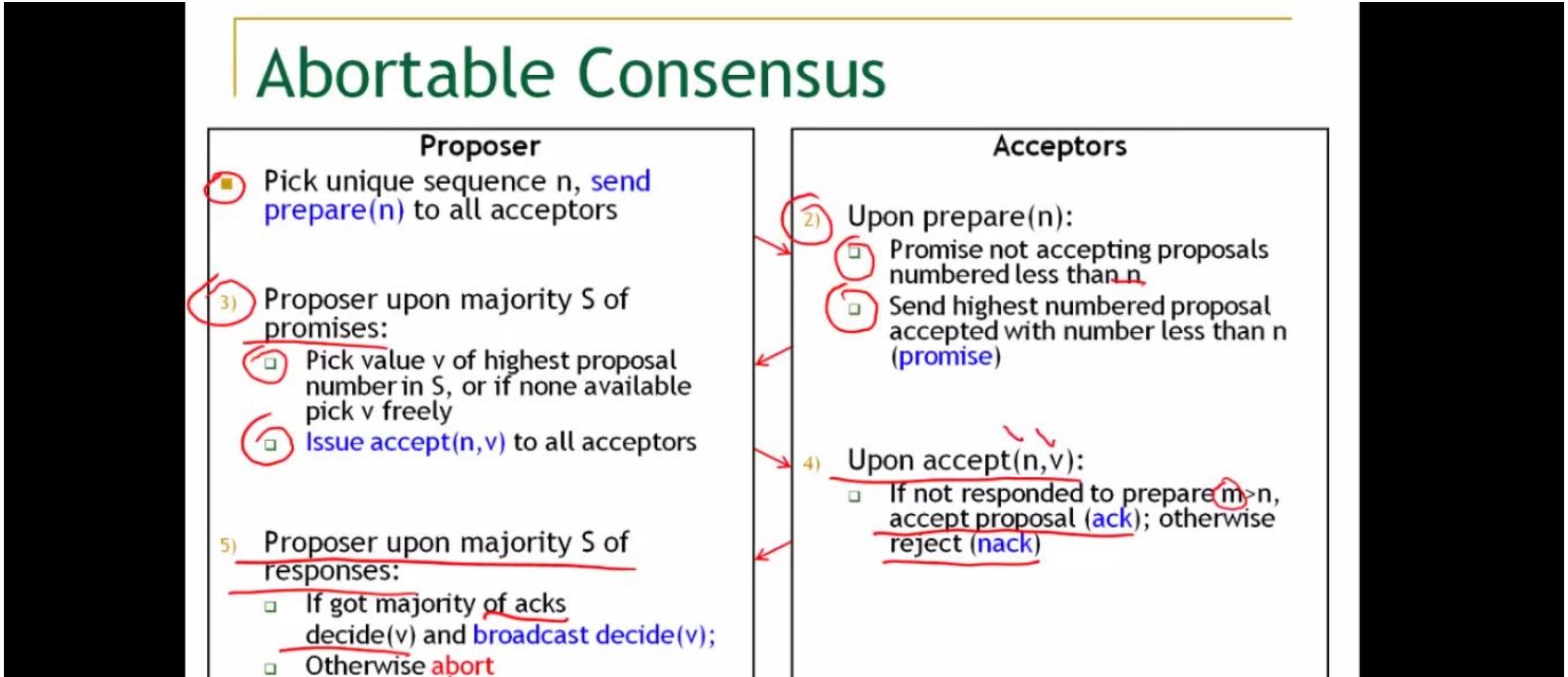
- We don’t know when the round ends :(

Asynchronous?

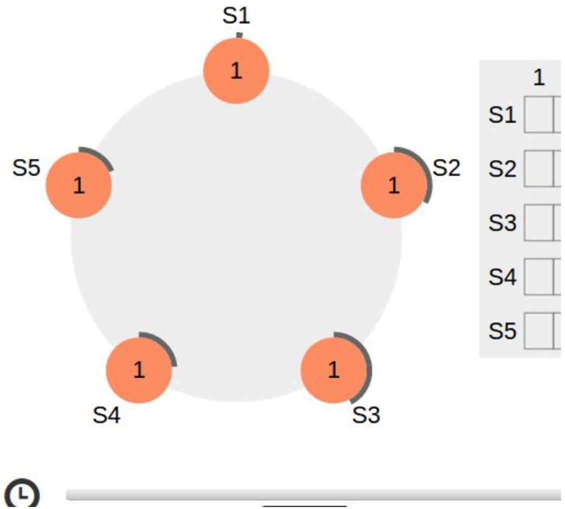
- We don't know when the round ends :(
- Majority Voting

Paxos

https://www.youtube.com/watch?v=WX4gjowx45E



Raft



https://raft.github.io

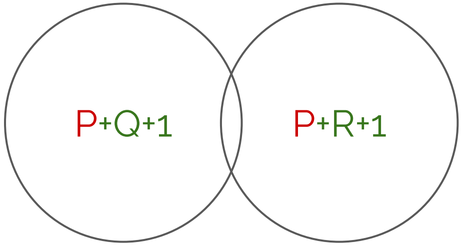
Byzantine Failures

ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

- Assume some nodes and the network may be actively malicious
  - They might not reply at all (direct DoS attack)
  - They might be able to prevent honest nodes from communicating (indirect DoS attack)
  - They might send different messages to different nodes (equivocation)
- Fundamentally need N=3f+1 for consensus in the general case
  - f out of N might not reply → Need to proceed with N-f or 2f+1
  - f out of the N-f might be malicious → Need majority
    - **N-2f > f => N>3f or N=3f+1**
- Can be relaxed to N=2f+1 under various stronger assumptions
  - Trusted hardware components to prevent equivocation
  - Assumptions that honest nodes can communicate within a finite time (synchrony)

Impossibility results

- No Byzantine consensus if **f >= N/3**
- Counter example: divide into 3 equal groups: P, Q and R
    - P is corrupted and contains the sender
    - Temporarily partition Q and R
    - P behaves as though the Sender says “0” and interacts with Q
    - P behaves as though the Sender says “1” and interacts with R



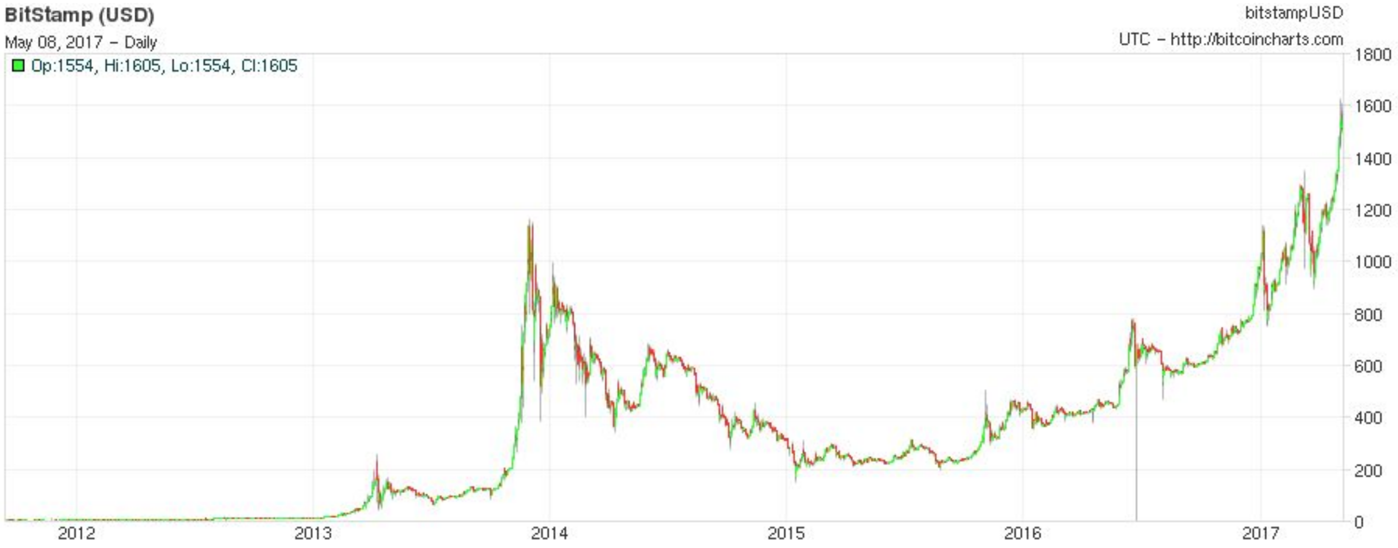
- (P and Q) must behave the same as if R has crashed (pick “0”)
- (P and R) must behave the same as if Q had crashed (pick “1”)

Outline

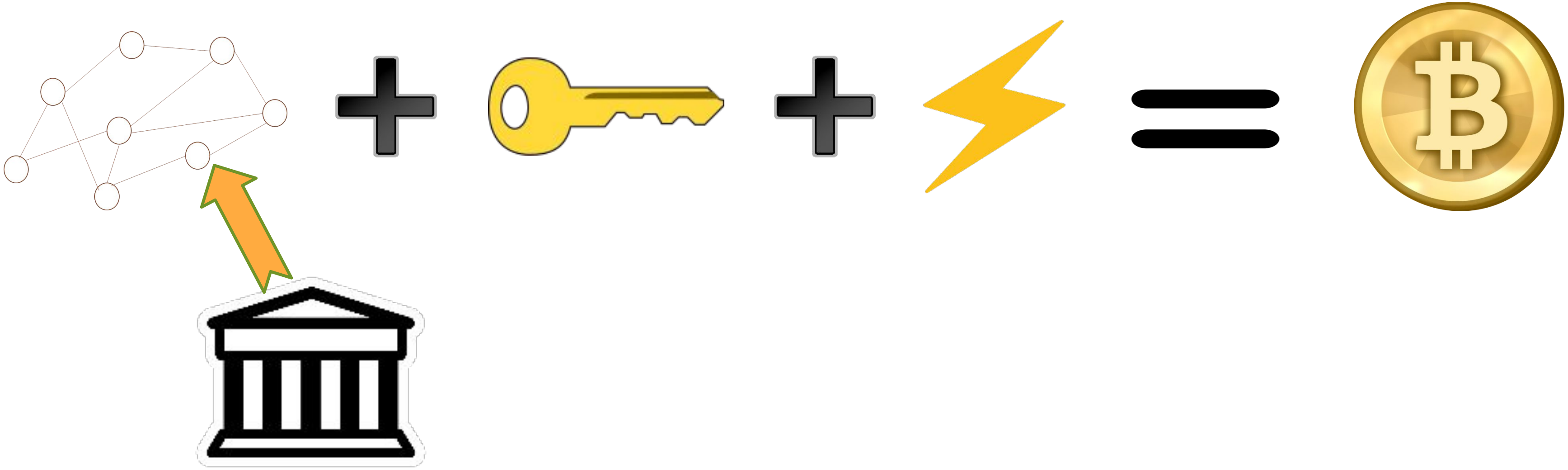
- Redundancy and Fault-Tolerance
- High Availability and Data Consistency
- Consensus
- **Bitcoin & Blockchains**

Bitcoin

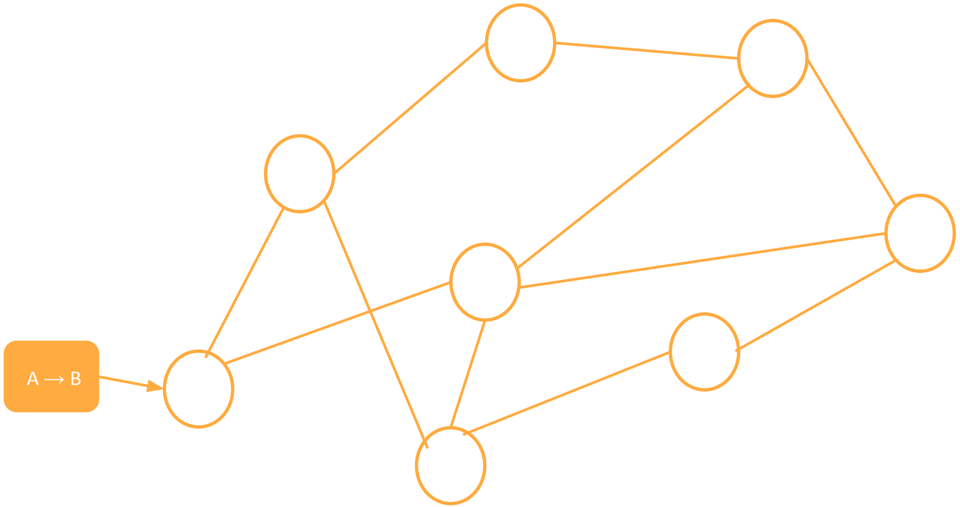
- Bitcoin is a cryptocurrency
  - Security based on asymmetric cryptography
  - Full client control over his currency



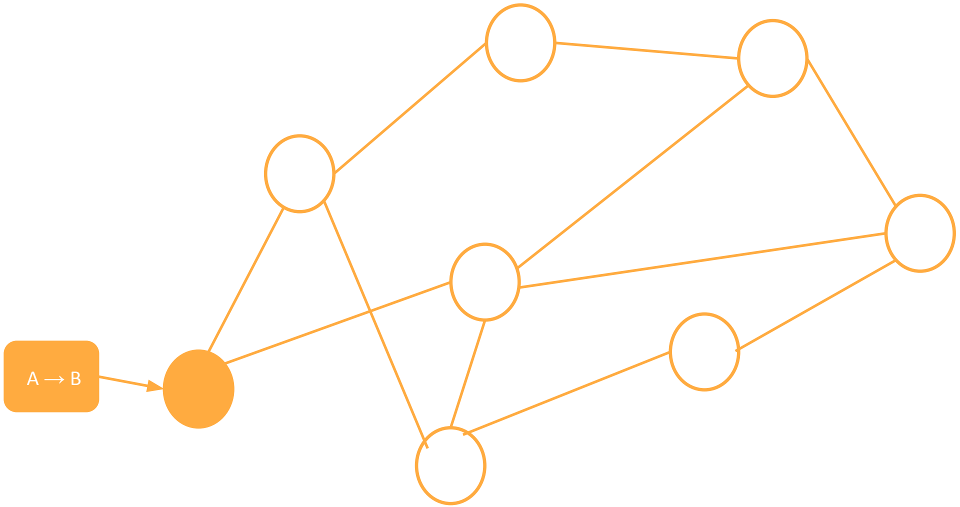
Bitcoin



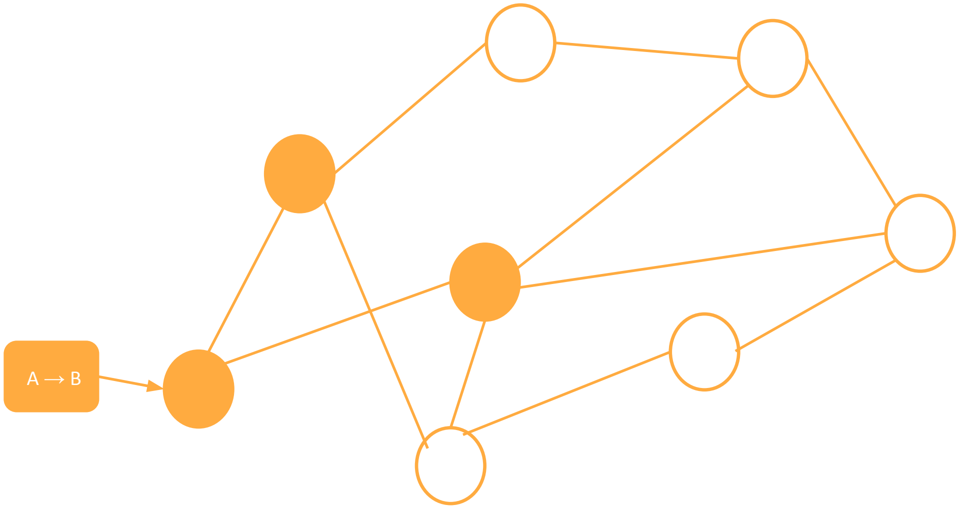
Transaction Verification in Bitcoin



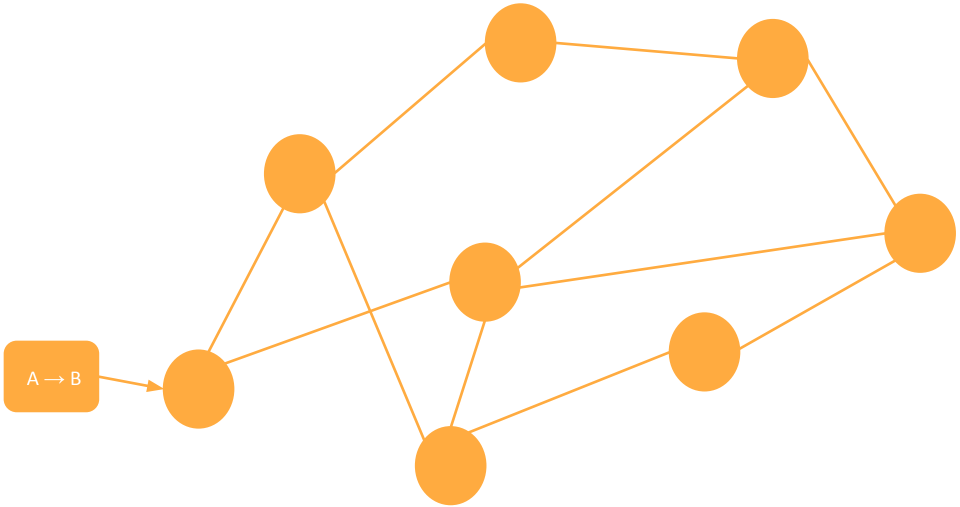
Transaction Verification in Bitcoin



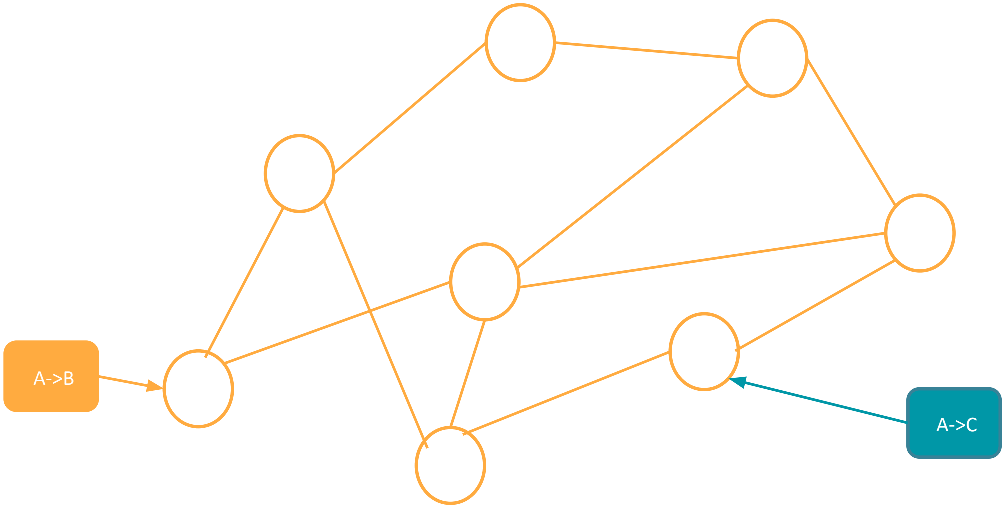
Transaction Verification in Bitcoin



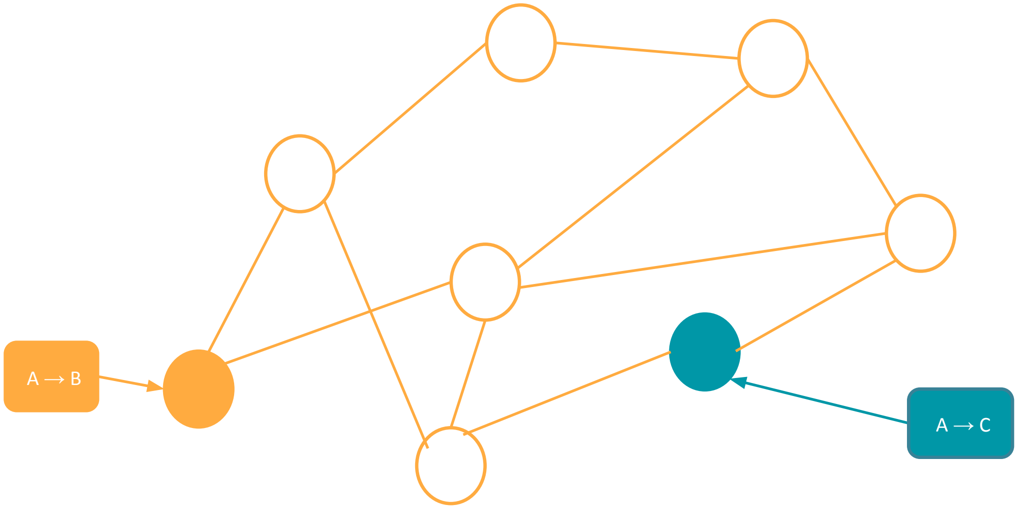
Transaction Verification in Bitcoin



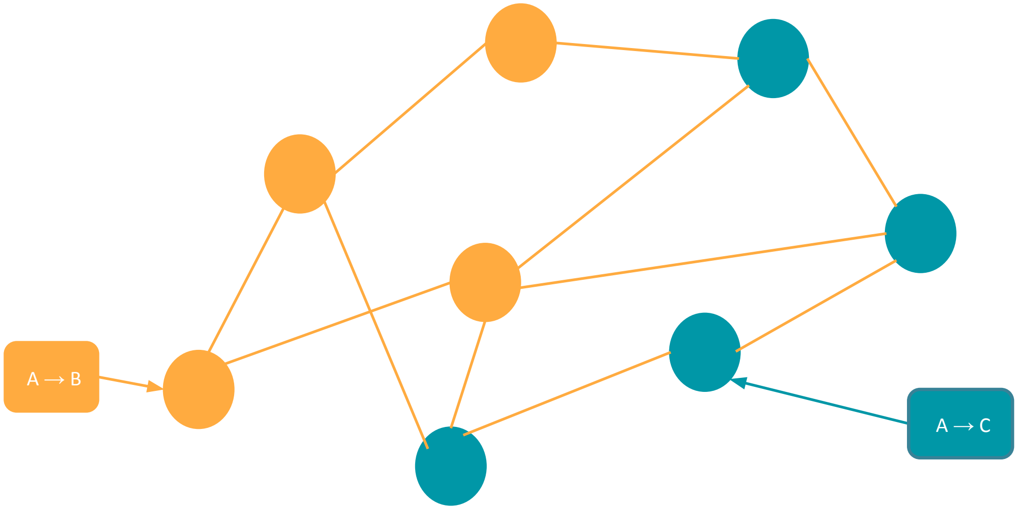
Conflict Resolution



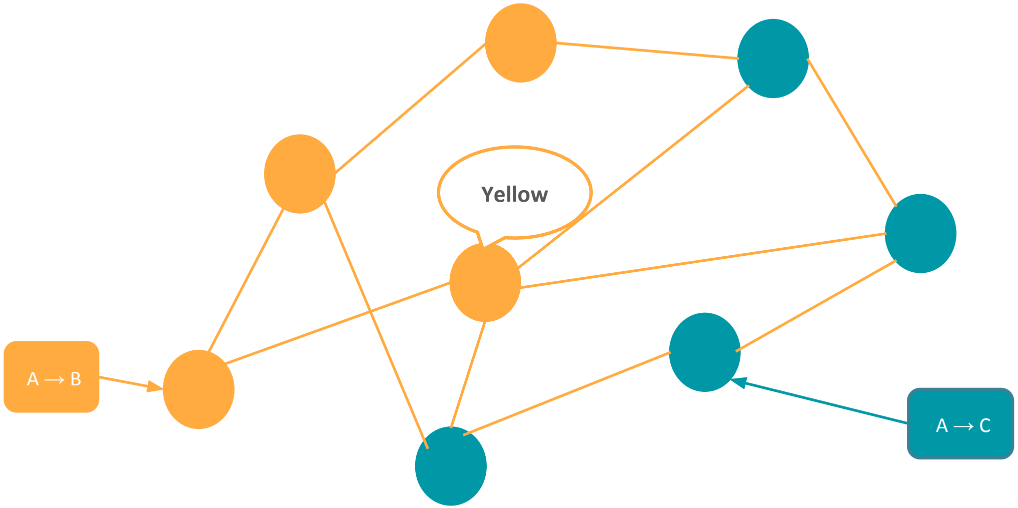
Conflict Resolution



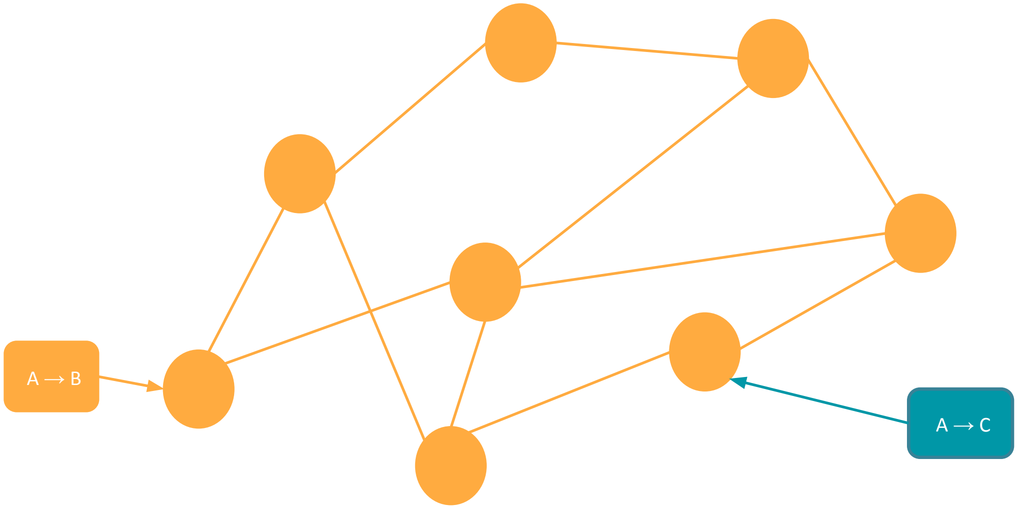
Conflict Resolution



Conflict Resolution



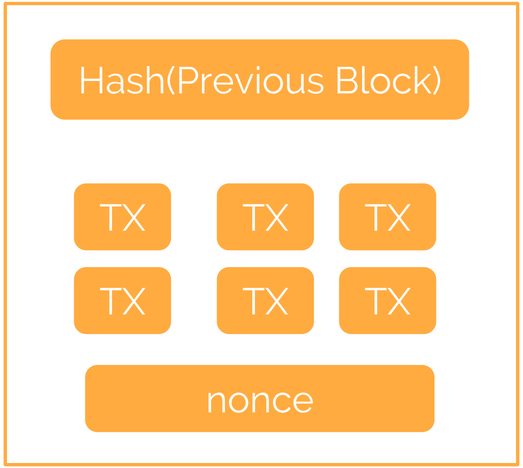
Conflict Resolution



Leader Election

Proof-of-Work

BLOCK



H(Block, nonce=0) =abc3426fe31233

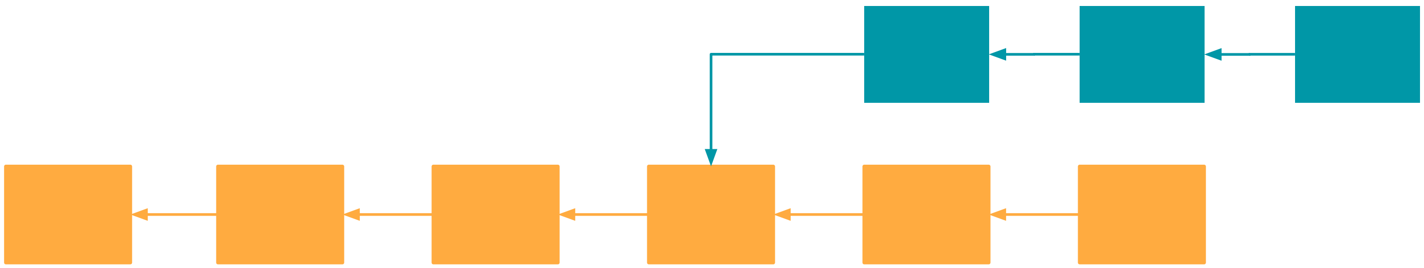
H(Block, nonce=1) =fe541200abc229

H(Block, nonce=2) =0bc3429831233

.
.
.
.

H(Block, nonce=f23) =0000fed98312

Unstable Consensus (Forks)



Question?

What happens if there is a network partition

a) The protocol halts preserving safety

b) Now we have 2 versions of Bitcoin that will never merge back

c) The clients do not realize it and can be attacked

d) Free money for everyone

ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE

Risk or Wait

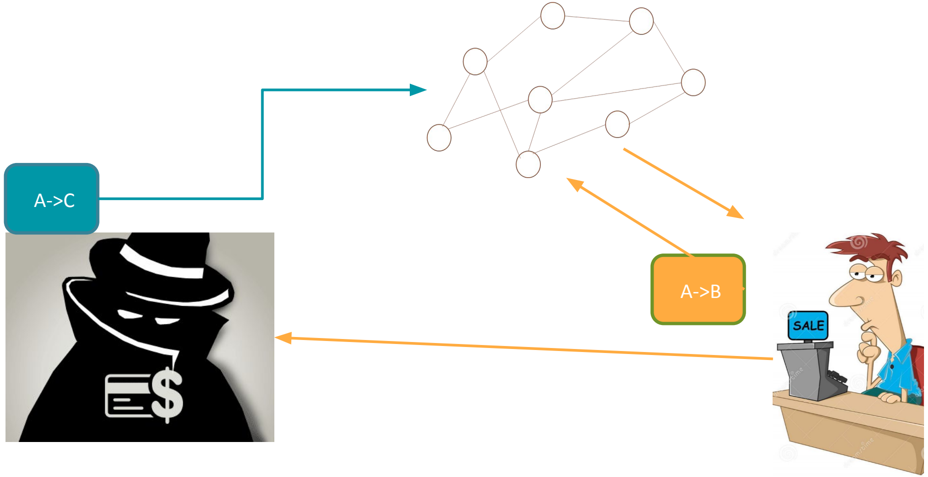
- In order for a transaction to be valid it needs to be confirmed by the blocks
    - Each confirmation takes **10 minutes**
    - Wait **one hour** to spend your money
    - Real time transactions are risky, **double-spending** them is not a hard thing to do

What's new about Bitcoin?

- We do not assume that we know all of the node IDs ahead of time!
    - This undercuts ~30 years of work.
- "Honest majority" measured as a fraction of "hashpower"
- Incentives for following the protocol (though this is an incomplete story)
- Nodes do not need to output a final decision (aka "stabilizing consensus")

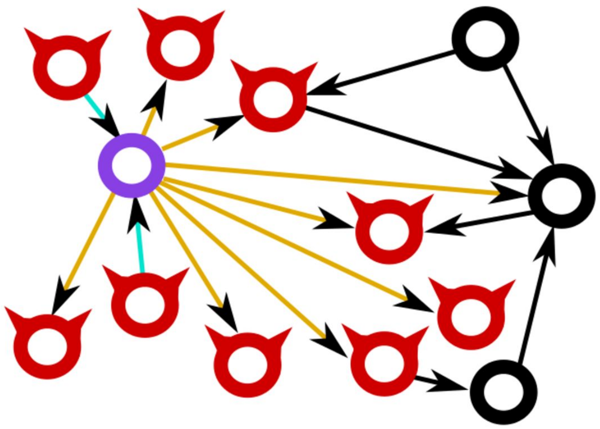
Double Spending Attack

1) Give transaction to seller
2) Take the product
3) Send a 2nd transaction and create a longer chain

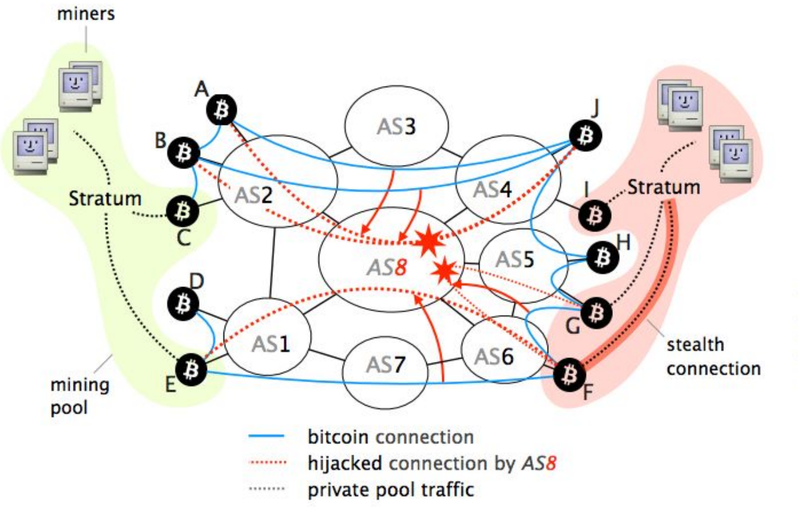


Is an AP system safe? Eclipsing

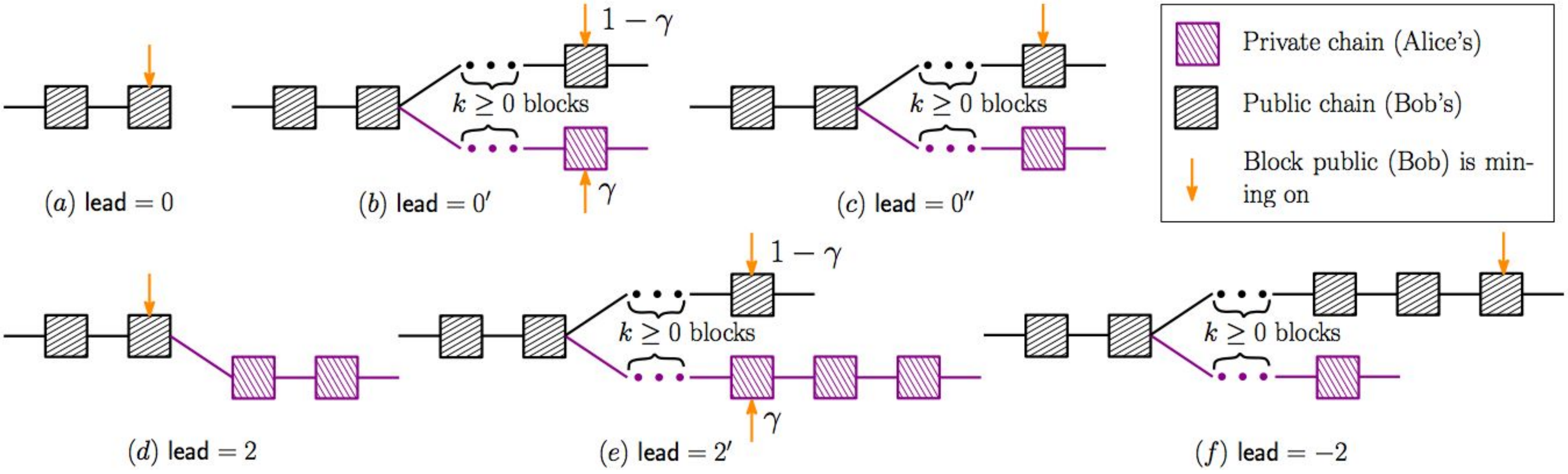
Eclipse Attacks on Bitcoin's Peer-to-Peer Network



Hijacking Bitcoin: Routing Attacks on Cryptocurrencies



Is an AP system safe? Strategic Mining



Acknowledgments

These slides are partly inspired by:

- CS-522 POCS EPFL
- Highly Available Transactions VLDB 2014
- ECE-598 AM UIUC
- CS426/526 Yale
- CS-451 Distributed Algorithms EPFL