CPSC 416 Distributed Systems

Winter 2023 Term 1 (November 9, 2023)

Tony Mason (fsgeek@cs.ubc.ca), Lecturer

Logistics

Teaching Assistants

Andy Hsu (andy.hsu@alumni.ubc.ca)

Hamid Ramezanikebrya (hamid@ece.ubc.ca)

Jonas Tai (jonastai@student.ubc.ca)

Cathy Yang (kaiqiany@student.ubc.ca)

Office Hours

Remember: Use Piazza for **all** official course-related communications

- Not on Piazza? Not official.
- Canvas "comments/messages" **are not monitored**

Office Hours:

Who	When	Where
Tony	Monday 14:00-15:00 Wednesday 16:00-17:00	Discord
Andy	Thursday 19:00-20:30	Discord
Hamid	Friday 16:30-18:00	Kaiser 4075
Jonas	Thursday 13:00-14:00	X241
Cathy	Friday 09:00-10:30	X237

Self-Assessment

This week

- DP3 Implementation Report (Today @ 23:59)

Next week

- No class (Tue 2023/11/14)
- Usual self-assessment activity (Thu @ 17:00)
- Capstone Week 5 Report (Thu @ 17:00)
- DP3 Implementation Report Peer Review (Thu @ 17:00)
- Capstone Project Team Declaration (Thu @ 17:00)

Note:

- You are strongly encouraged to collaborate with others on this
- You should use tools at your disposal to answer these questions
- **Do not forget to submit it.**

Readings

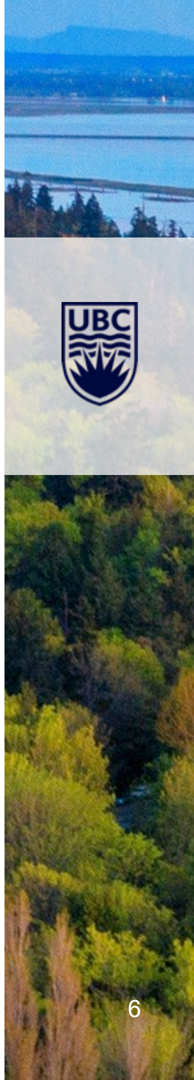
Required:

The Byzantine Generals Problem (Lamport/Shostak/Pease, TOPLOS 1982)

Practical Byzantine Fault Tolerance (Castro/Liskov, OSDI 1999)

Recommended:

Making Reads in BFT State Machine Replication Fast, Linearizable, and Live (2021)



Today’s Failure

Linux Kernel Bug

Date: June 30, 2012

Time: 23:60 UTC

Source: Wired

International Telecommunications Union (ITU) added one second to the clock (a *leap second*)

Impact: Reddit, LinkedIn, Quantas Airlines Reservations failed (plus many others)

Bug: Linux kernel

Root cause: bug in the clock logic caused “thundering herd” (waking all threads up) and the massive CPU load caused cascading failures.

Lesson Goals

Byzantine Fault Tolerance

Byzantine Systems

Practical Byzantine Fault Tolerance

Blockchain

Introduction: Byzantine Fault Tolerance

Consensus *with* Byzantine failures

Practical Byzantine Fault Tolerance (pBFT)

Blockchain: Byzantine proof distributed consensus

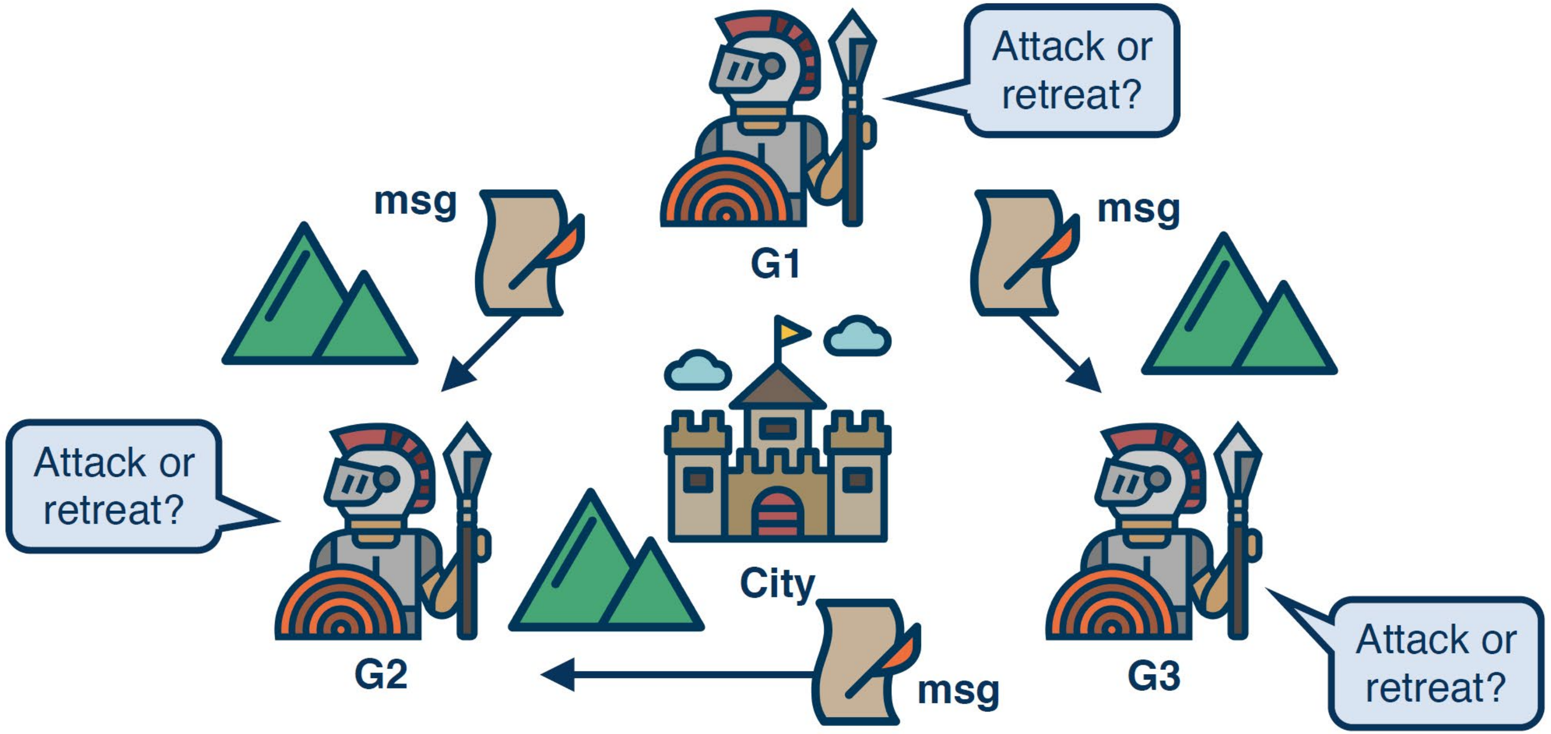
Byzantine Failures

Model change: Nodes **continue to participate** after failure

- Could be *malicious*
- Incorrect behaviour: incorrect messages

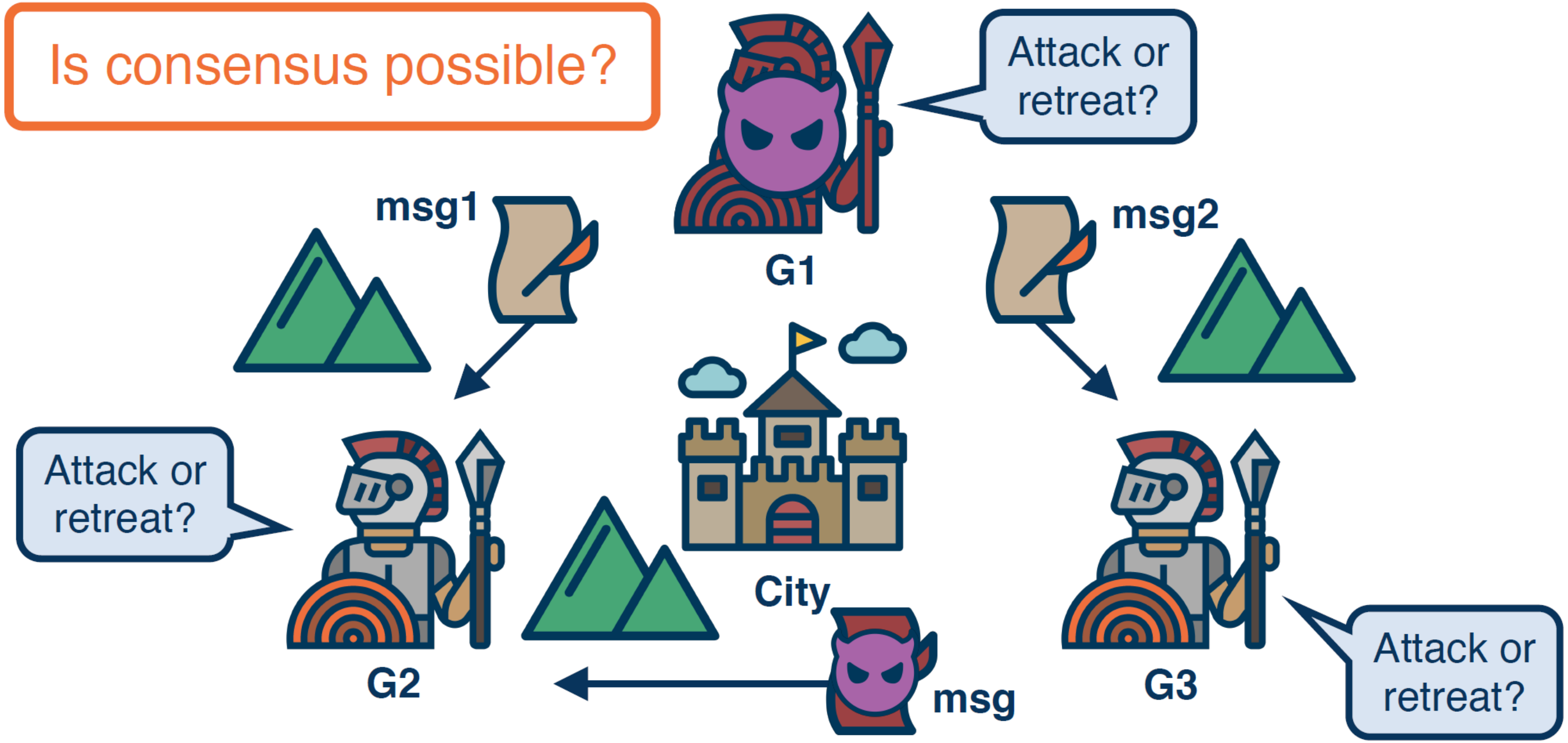
The Byzantine Generals Problem

Byzantine Generals Problem



Byzantine Generals Problem

Is consensus possible?



Goals of Byzantine Fault Tolerance

Achieve consensus

- Safety
- Liveness
- Validity

Tolerate *f* failures

Asynchronous network

Allow **Byzantine** behaviour

How?

Messages

- Cryptographic signatures

Malicious participants

- Increase number of total participants
- For *f* faults: need *3f + 1* nodes

Corrupt Leader

- Add checks among participants

Liveness: bounded delay (“eventual synchrony”)

Practical Byzantine Fault Tolerance

Practical Byzantine Fault Tolerance (Castro & Liskov, OSDI 1999)

High performance

- Tolerates *f* failures with *3f* + 1 nodes
- 97% as fast *with* replication (using NFS)

pBFT: System Model

Replicated Service

- *3f* + 1 replicated nodes (for up to *f* failures)
- Primary + replicas

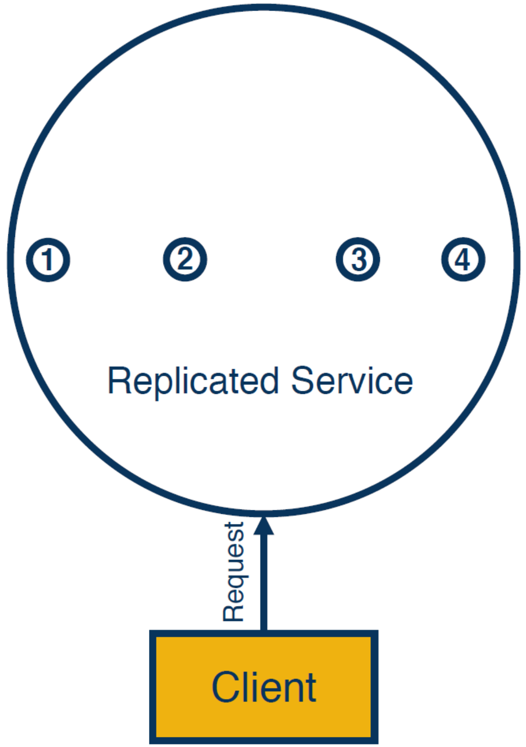
Uses a *view* defined by current primary

Replicas are replicated state machines

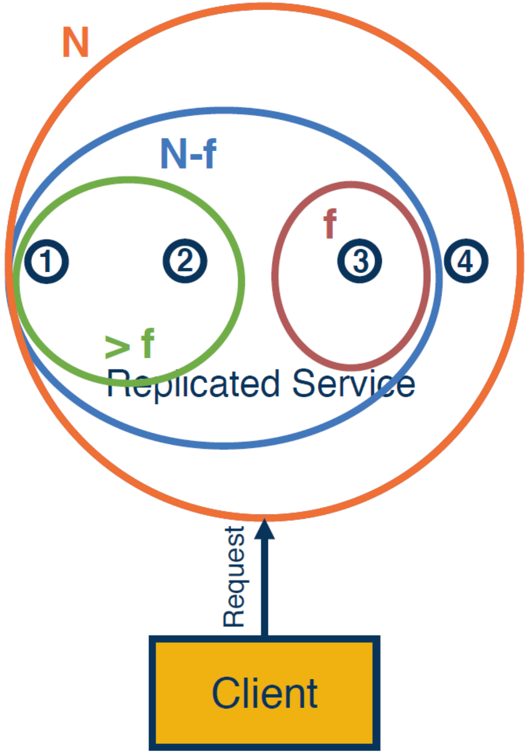
- Consistent
- State includes: service state, message log, current view

Communications integrity

- Digests
- Public keys



Why we need *3f + 1*



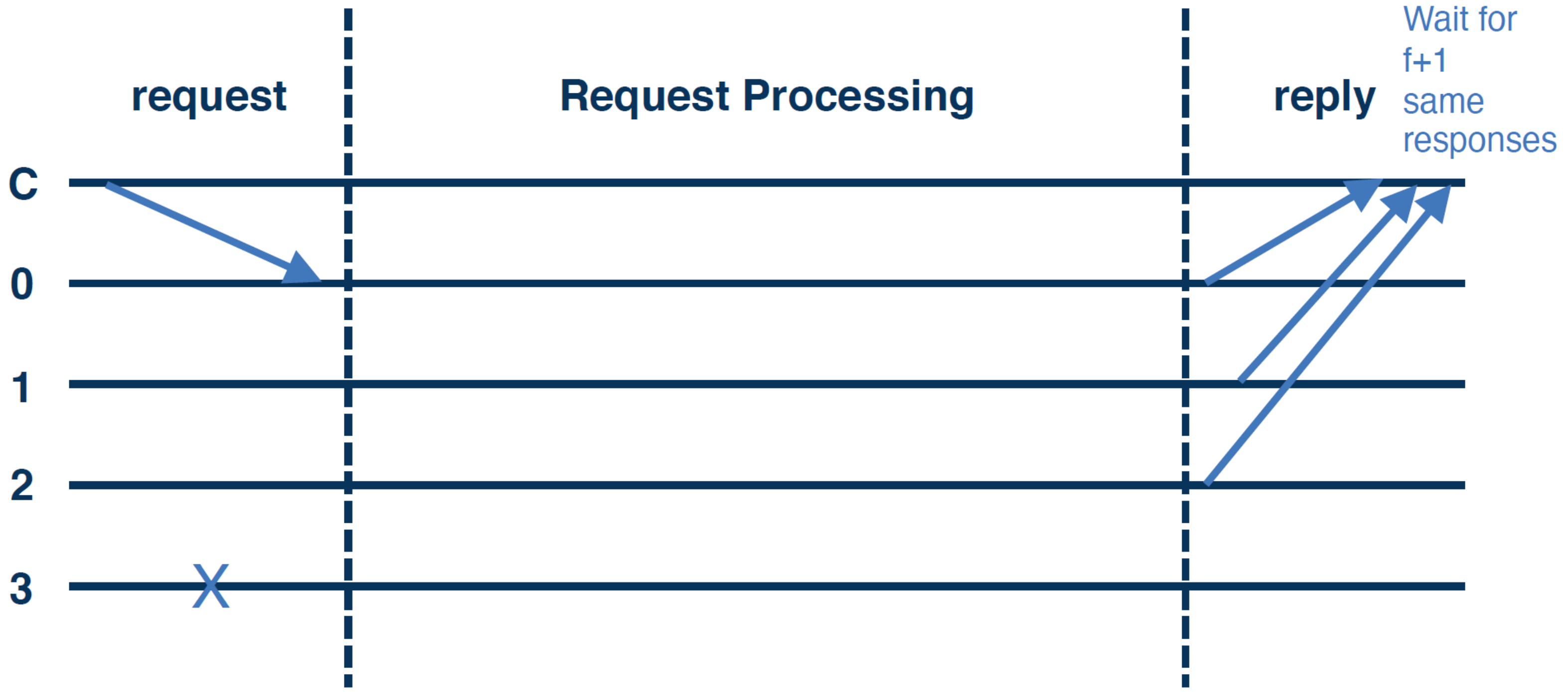
N nodes

Permit *f* faults (including Byzantine)

Requires quorum among *N-f* nodes

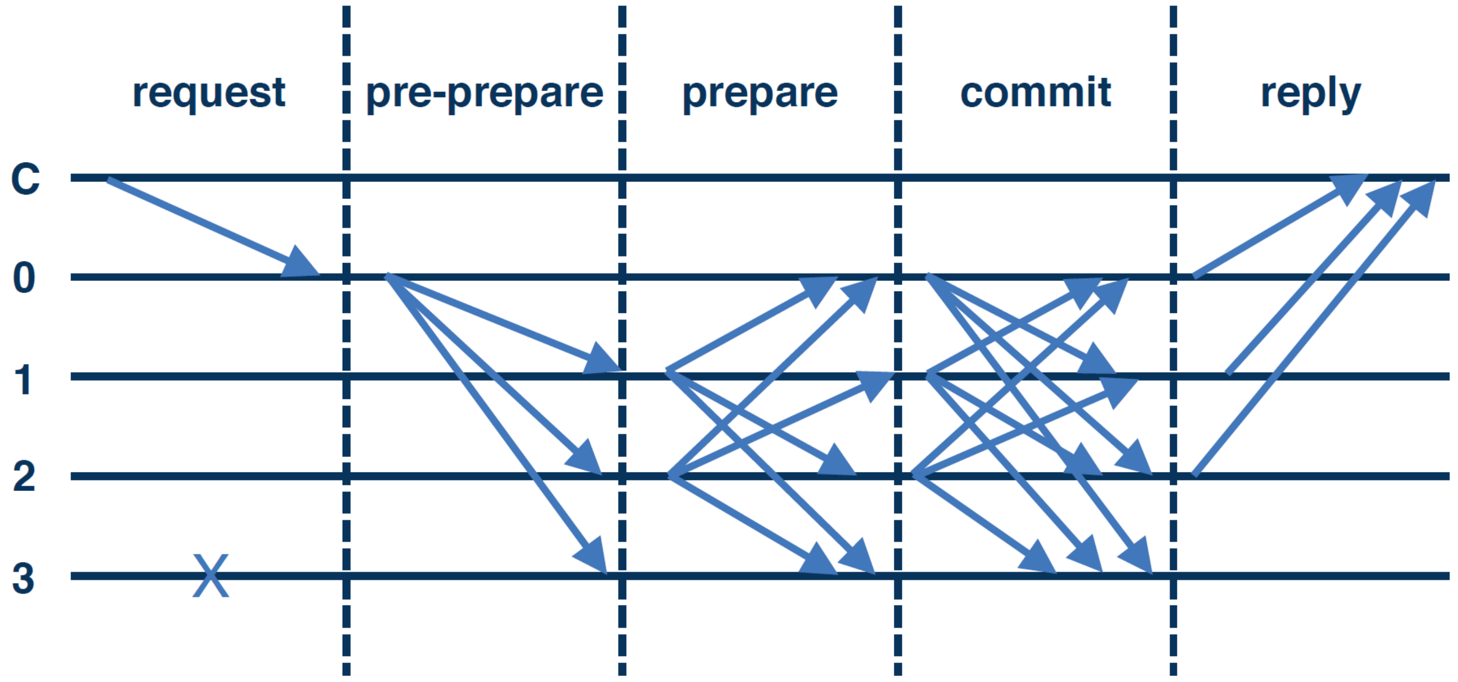
$N > 3f$ (e.g., $3f + 1$)

pBFT: Request Processing



Adapted from: http://pmg.csail.mit.edu/papers/osdi99.pdf

pBFT: 3PC protocol



Adapted from: http://pmg.csail.mit.edu/papers/osdi99.pdf

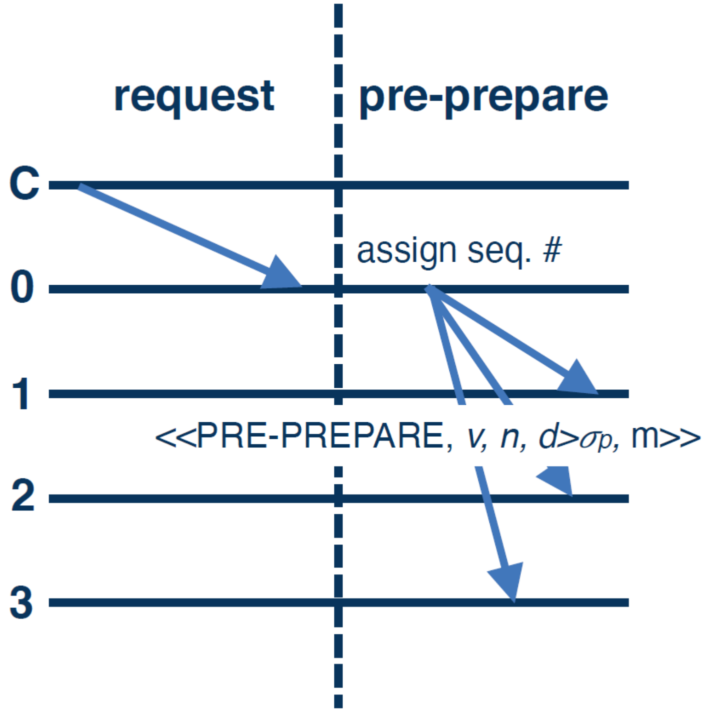
pBFT: Pre-Prepare Phase

Leader multicasts pre-prepare request *with the message to the backups*.

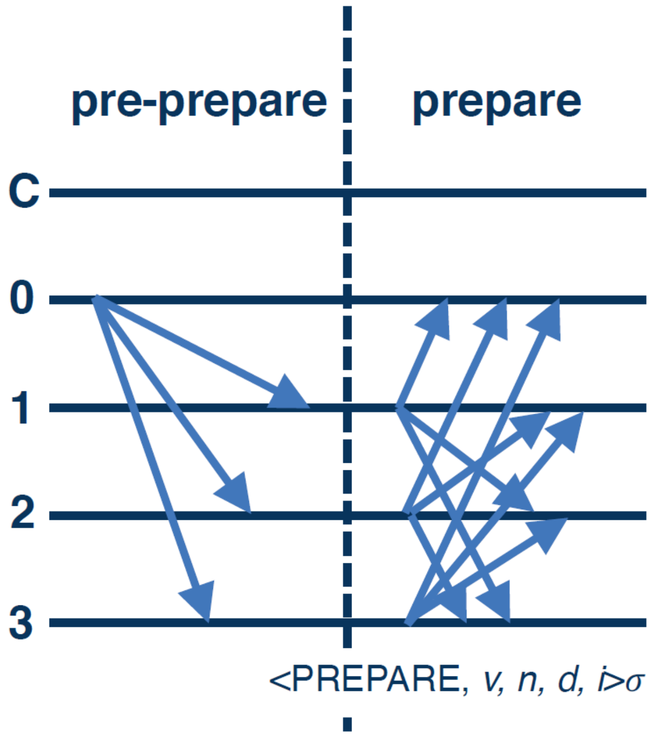
Leader records message in its log

Replicas accept pre-prepare *if*:

- Signature σ and digest ω check
- View ν is correct
- Sequence number μ is new
- μ such that $\rho < \mu < \mathrm{P}$. These are the *watermarks*



pBFT: Prepare Phase



At least one replica multicasts a *prepare* message (after accepting *pre-prepare*)

Waits for consensus responses

- prepared messages
- Log contains pre-prepare and 2f matching prepare
  - Same view
  - Same sequence number
  - Same digest

pBFT: Commit Phase

Replica multicasts *commit* message (after prepare)

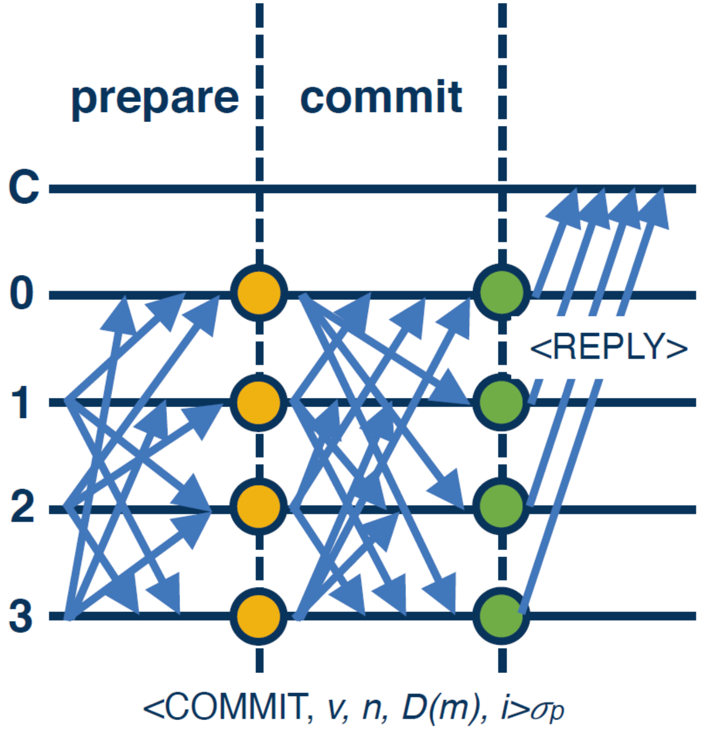
Waits for responses

- Committed

Commit when

- Prepared is true
- 2f + 1 matching committed messages seen (including replica)

Can reply to client once commited



pBFT: More Details (in Paper)

Garbage collection (log)

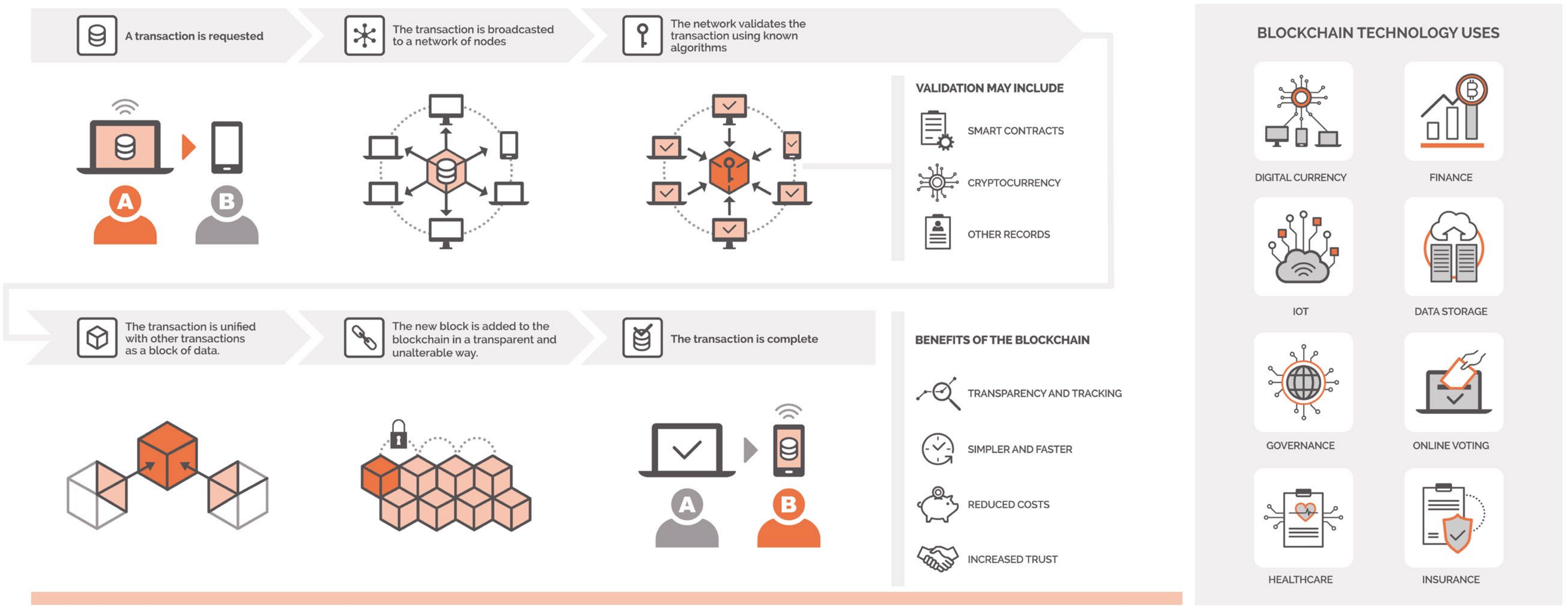
View changes

Liveness

Performance optimizations (message elimination)

Sample Byzantine-fault tolerant service (replicated NFS)

HOW DOES BLOCKCHAIN WORK



1. A transaction is requested
2. The transaction is broadcasted to a network of nodes
3. The network validates the transaction using known algorithms
4. The transaction is unified with other transactions as a block of data.
5. The new block is added to the blockchain in a transparent and unalterable way.
6. The transaction is complete

VALIDATION MAY INCLUDE

- SMART CONTRACTS
- CRYPTOCURRENCY
- OTHER RECORDS

BENEFITS OF THE BLOCKCHAIN

- TRANSPARENCY AND TRACKING
- SIMPLER AND FASTER
- REDUCED COSTS
- INCREASED TRUST

BLOCKCHAIN TECHNOLOGY USES

- DIGITAL CURRENCY
- FINANCE
- IOT
- DATA STORAGE
- GOVERNANCE
- ONLINE VOTING
- HEALTHCARE
- INSURANCE

Bitcoin

Bitcoin: A Peer-to-Peer Electronic Cash System (Nakamoto, 2008)

Byzantine system: the “double spend”

Basic unit is the transaction block:

- Balanced set of operations
- Public
- Easily verified

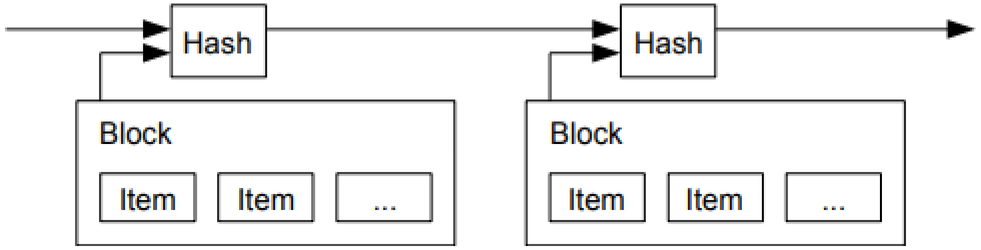
Implements a *distributed timestamp service*

Blockchain: the *chain*

A cryptographic hash is computed for each block of information.

New hash: previous block hash + hash of current block

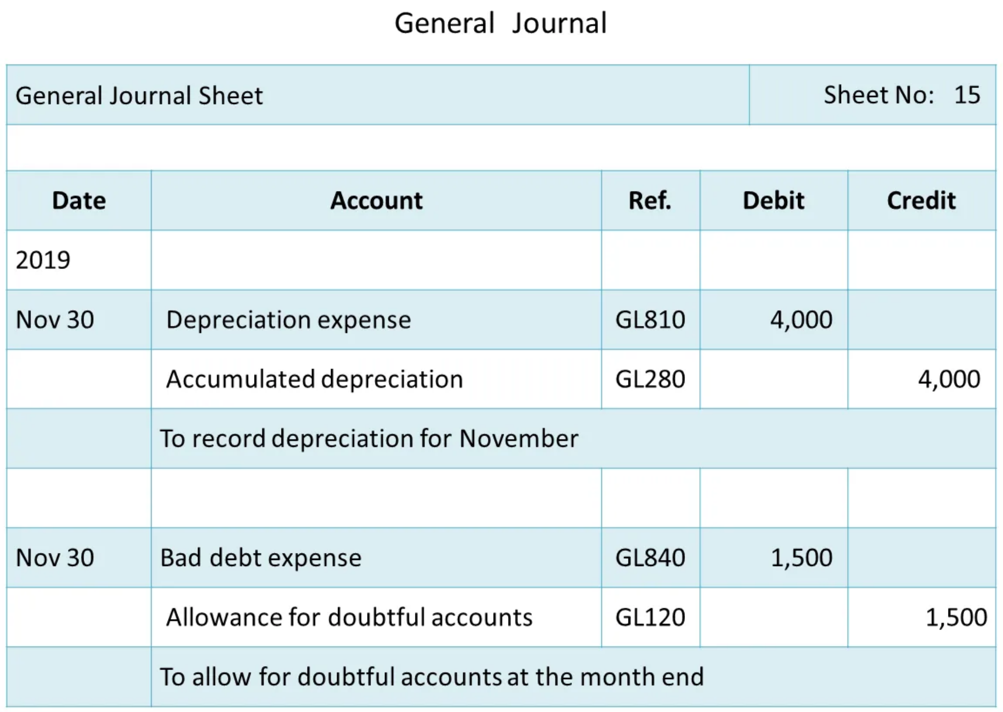
- Creates the *chain*
- Makes it difficult to "rewrite history"



Blockchain: Ledger

Accounting 101

- Inflows = Outflows



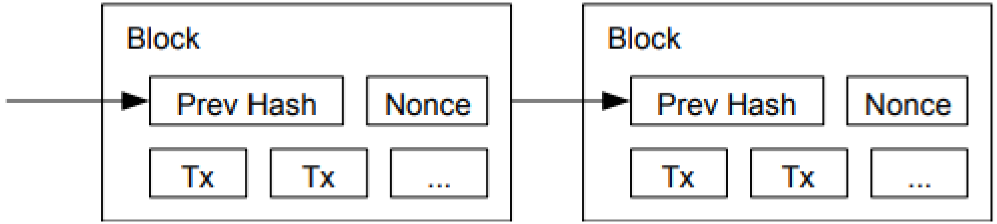
General Journal

General Journal Sheet				Sheet No: 15
Date	**Account**	**Ref.**	**Debit**	**Credit**
2019				
Nov 30	Depreciation expense	GL810	4,000	
	Accumulated depreciation	GL280		4,000
	To record depreciation for November			
Nov 30	Bad debt expense	GL840	1,500	
	Allowance for doubtful accounts	GL120		1,500
	To allow for doubtful accounts at the month end			

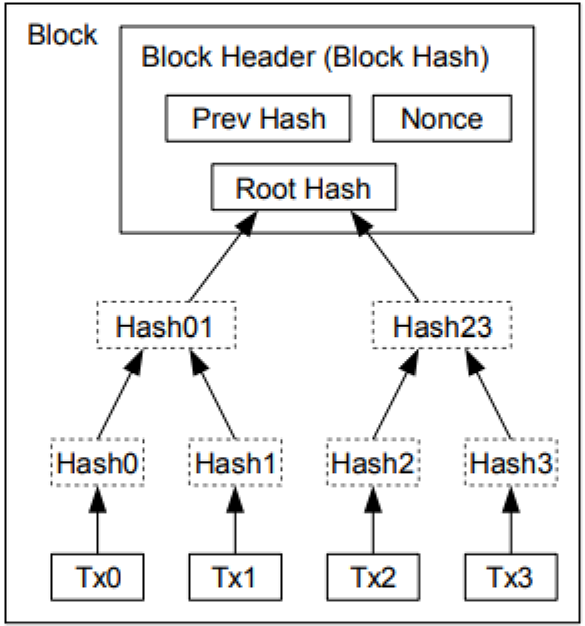
Blockchain: Proof of Work

Combine hash with a *nonce*

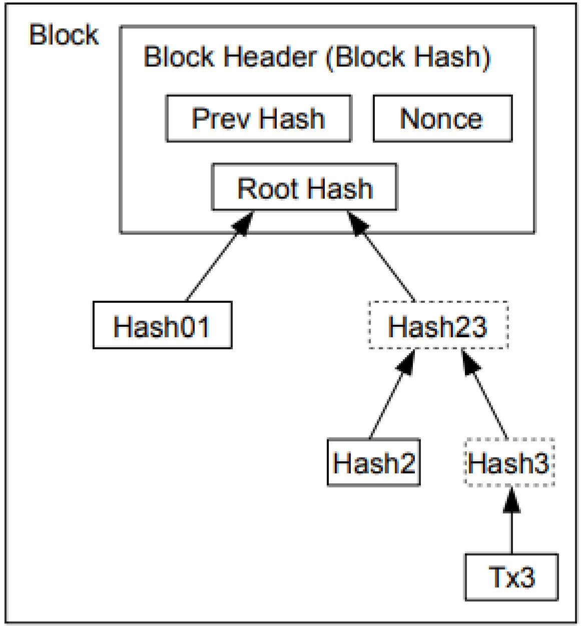
- Nonce is a value chosen so the hash has a specific number of zero bits (the *difficulty*)
- **Only way to find a nonce is to compute the hash**



Blockchain: Garbage Collection



Transactions Hashed in a Merkle Tree



After Pruning Tx0-2 from the Block

Blockchain vs pBFT

✓ Advantages

- Decentralized consensus
- Byzantine failures
- Unreliable network

✗ Negatives

- Tolerate f faults in N=3f+1 nodes!?
- Communication costs O(n^3)

Blockchain versus Bitcoin

Byzantine consensus for timestamped chained ledger blocks

- Not explicit in Nakamoto’s paper

Limits to participation

- Miners: must be willing to expend energy for Proof-of-work
- Cryptography

Incentivize good behavior

- Most participants want the product
- Economic factors discourage dishonesty (miners get rewards)

Additional Readings

Algorand: Scaling Byzantine Agreements for Cryptocurrencies
(Gilad/Hemo/Micali/Vlachos/Zeldovich, SOSP 2017)

Algorand: the Defi company

Ethereum Proof-of-Stake

- Lower energy consumption
- Consensus based upon *ownership* (ergo “weighted quorum”)
- Non-fungible tokens (NFT)

A Blockchain-based Land Title Management System for Bangladesh

Lesson Review

Byzantine Fault Tolerance

Byzantine Systems

Practical Byzantine Fault Tolerance

Blockchain

Questions?

THE UNIVERSITY OF BRITISH COLUMBIA