CS 5594: BLOCKCHAIN TECHNOLOGIES

Spring 2024

THANG HOANG, PhD

DISTRIBUTED SYSTEMS

Outline

Definition and Characteristics

Architecture Models

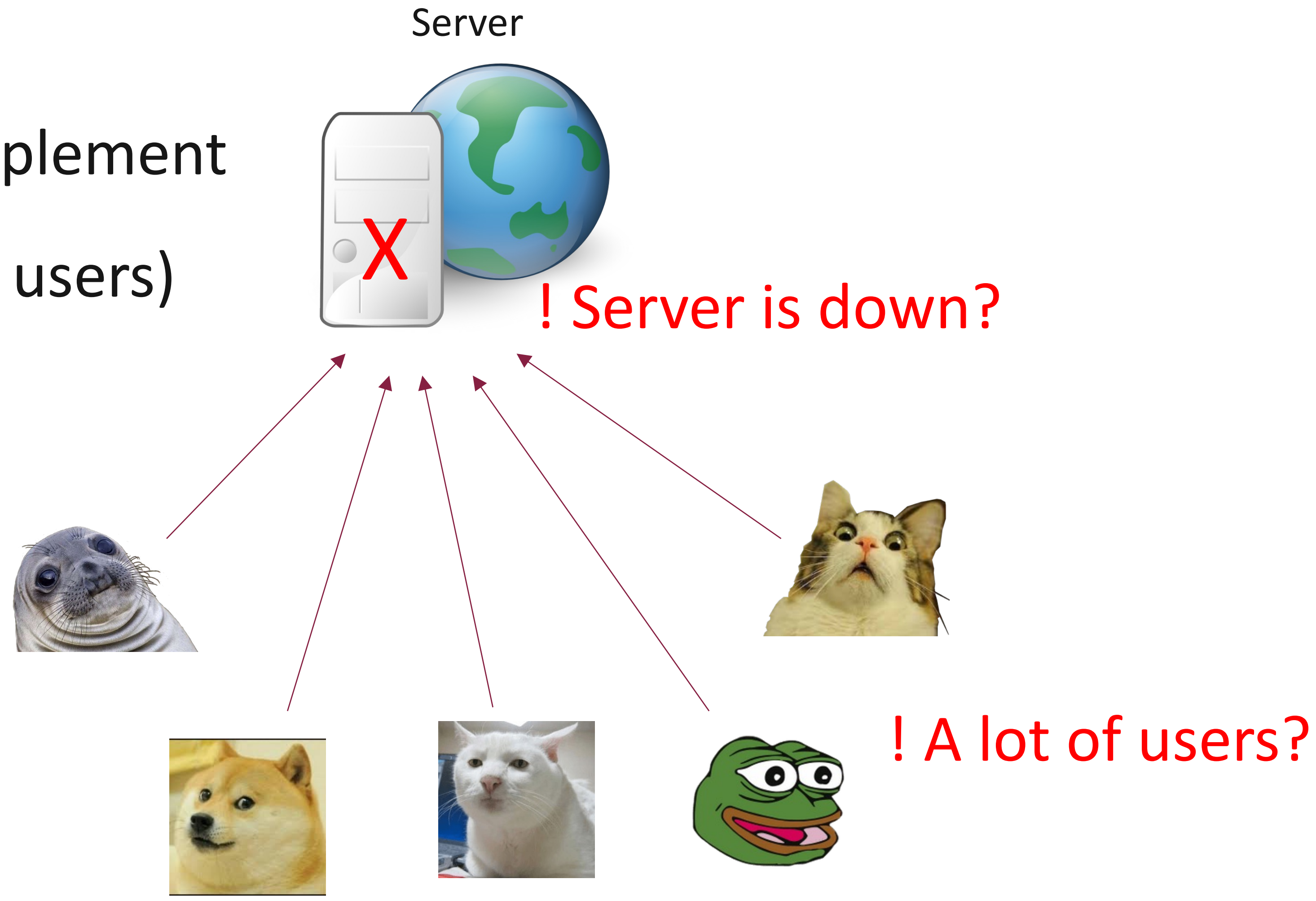
Distributed Algorithms

Definition and Characteristics

Centralized Systems

All tasks completed by a **single** entity

- ✓Simple
- ✓Easy to design and implement
- ✓Efficient (with small # users)



Distributed Systems

A group of independent entities communicated with one another in a coordinated manner

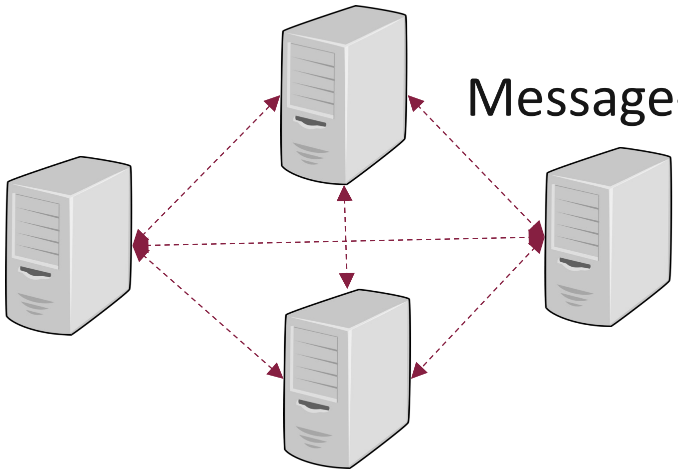
Collaboratively enable a service (computing, data sharing and storage)

Goal: Address inherent limitations of centralized systems

- Robustness
- Scalability
- Reliability

Distributed Systems

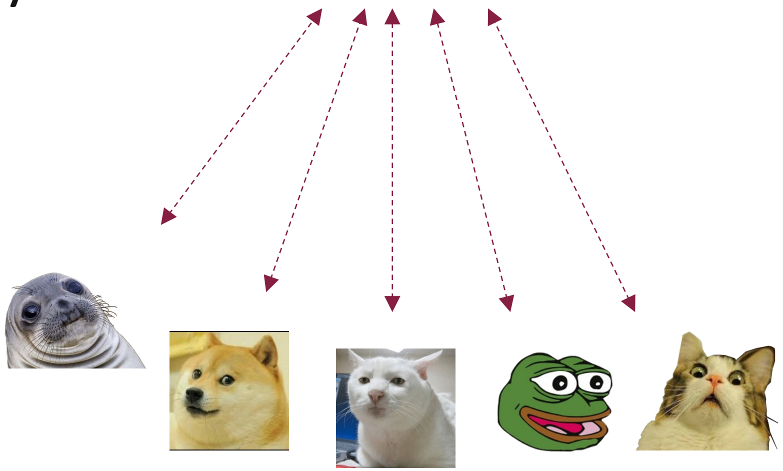
Distributed server



Message-based communication

Each entity has separate resource (data, memory, processor, OS)

Appeared as a single entity



Key Characteristics

Transparency

Most important feature

Illusion of a single system

- Hide all internal organization, communication details
- Uniform interface

Access transparency, location transparency, relocation transparency, migration transparency, replication transparency, concurrency transparency, failure transparency, scaling transparency, performance transparency

Key Characteristics

Openness

Heterogeneity

- Variety and differences in hardware and software components

Resource Sharing

- Resources (hardware, software, data) accessed across multiple entities

Concurrency

- Parallel executions of activities
- Reduce latency, increase throughput

Key Characteristics

Scalability

Add/remove components to/from the system

Fault Tolerance

Continuous availability

Design Goals

High Performance

Low latency, high throughput

Reliability

Preserve correctness and integrity in the presence of faulty/malicious nodes

Failure detection, self-stabilization

Scalability

Adapt with flexible number of users in the system

Design Goals

Consistency

Update consistency, replication consistency, cache consistency, failure consistency, clock consistency, user interface consistency

Synchronization between concurrent tasks

Security

Malicious adversaries, secure communication, resource protection

Impossibility Result

CAP Theorem

"Any distributed system cannot achieve Consistency, Availability and Partition tolerance concurrently."

Gilbert and Lynch

Impossibility Result

Consistency

All nodes see the same data at the same time

Availability

If the node in the system does not fail, it must always respond to the user's request.

Partition tolerance

The network will be allowed to lose arbitrarily many messages sent from one node to another

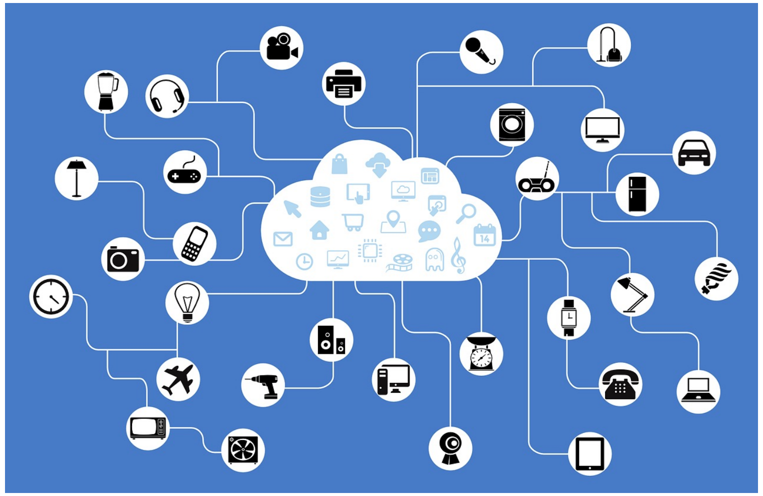
Choose 2 out of 3

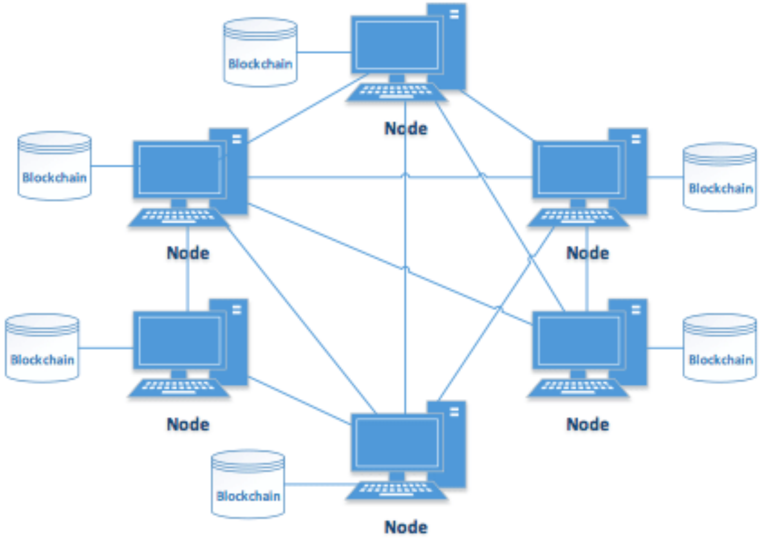
Generally between consistency & availability under partition

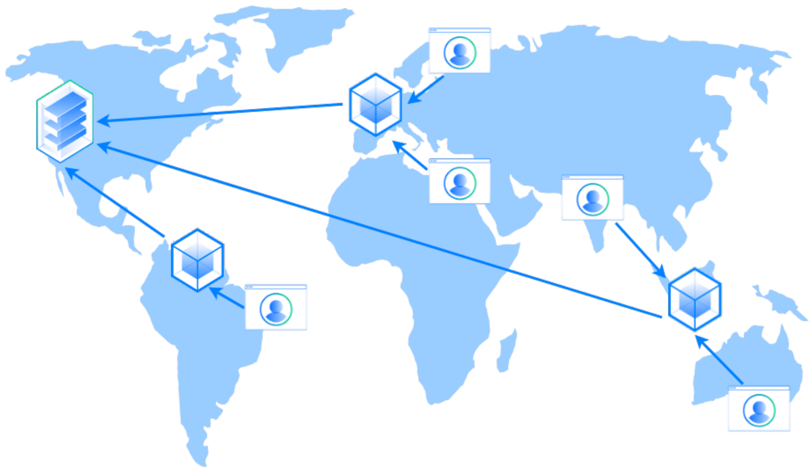
Distributed System Applications

Distributed systems are everywhere

- Mobile systems
- Sensor networks
- IoT
- Ubiquitous and Pervasive computing
- WWW
- P2P computing







Content Delivery Network (CDN)

Architectural Models

Client-Server Architecture

Basic model

Two types of node: client (slave) and server (master)

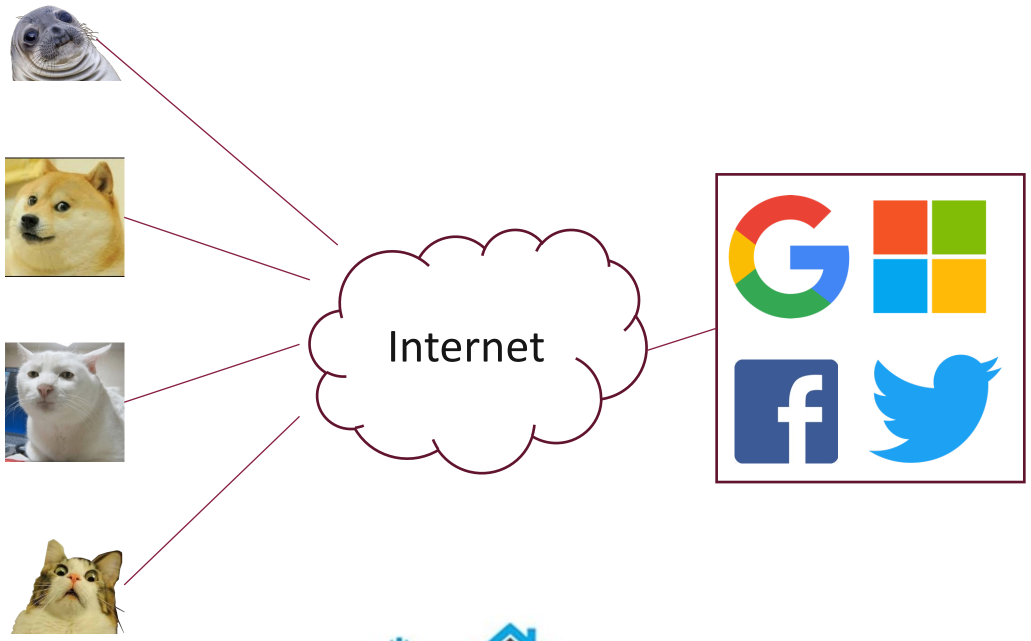
All tasks accomplished by server

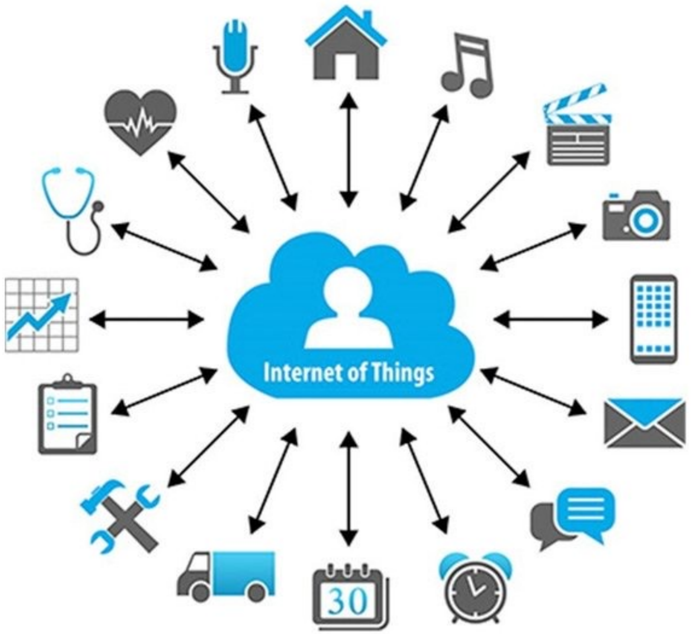
Server is resource-powerful

Client is resource-limited

Asymmetric, partially distributed

<u>Examples:</u> Cloud services (Amazon, MS, Facebook, Google), IoTs





Client-Server Architecture

Advantage

Easy to maintain security and reliability

Enable a wide range of services

Easy to design and implement

Client-Server Architecture

Disadvantage

- Central point of failure and compromise
  - Attacks targeting to server nodes (e.g., DoS, data-breach)
- Resource management and administration
- Central point of trust
  - Server has more control and authority in the system
- Not so scalable
  - More clients join, more server demands

Peer-to-Peer Architecture

A network of nodes (peers) sharing resources directly with each other

Symmetric: All nodes are equal participants and play both roles: provider and consumer of resource

No *server* node

Fully distributed, no centralized data and resource

“The ultimate form of democracy on the Internet”

Examples: blockchains, vehicular network, file-sharing

Bitcoin: A Peer-to-Peer Electronic Cash Systen

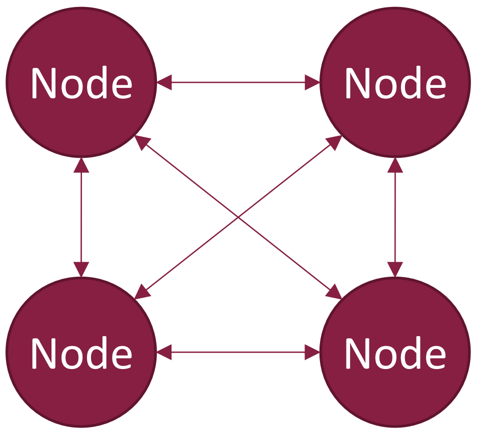
Satoshi Nakamoto
satoshin@gmx.com
www.bitcoin.org

Abstract. A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network.

Peer-to-Peer Architecture

Advantage

- Distributed trust
- Balanced resource load
- High resource capacity and high scalability
  - More clients, more servers
- High fault-tolerance and resiliency against DoS attacks



Peer-to-Peer Architecture

Disadvantage

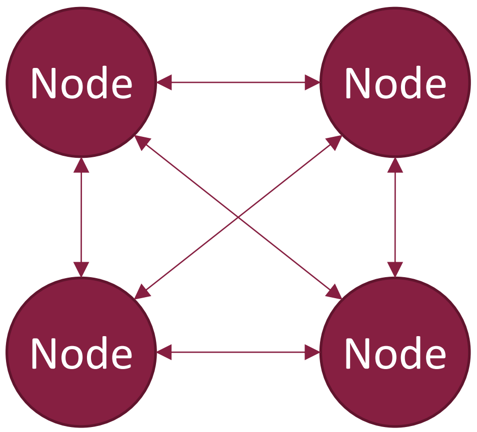
Costly backup, high bandwidth consumption

Hard to control

Hard to maintain security and consistency

Vulnerable to network partitions, byzantine behavior

Unstable



Distributed vs. Decentralized

P2P is distributed, but offers various degrees of decentralization

Some P2P still need central authorities to make decision (e.g., network control, resource load) efficiently

- Somewhat centralized

Decentralized is NOT all-or-nothing

Distributed vs. Decentralized

In fact, no system is purely decentralized, or purely centralized

Blockchain can be centralized or decentralized under certain degrees

- Depend on the design and application requirements

Unstructured P2P network

Easy to build

Loose restriction on overlay structure, data location and resource distribution

- Nodes communicate randomly, perform arbitrary tasks

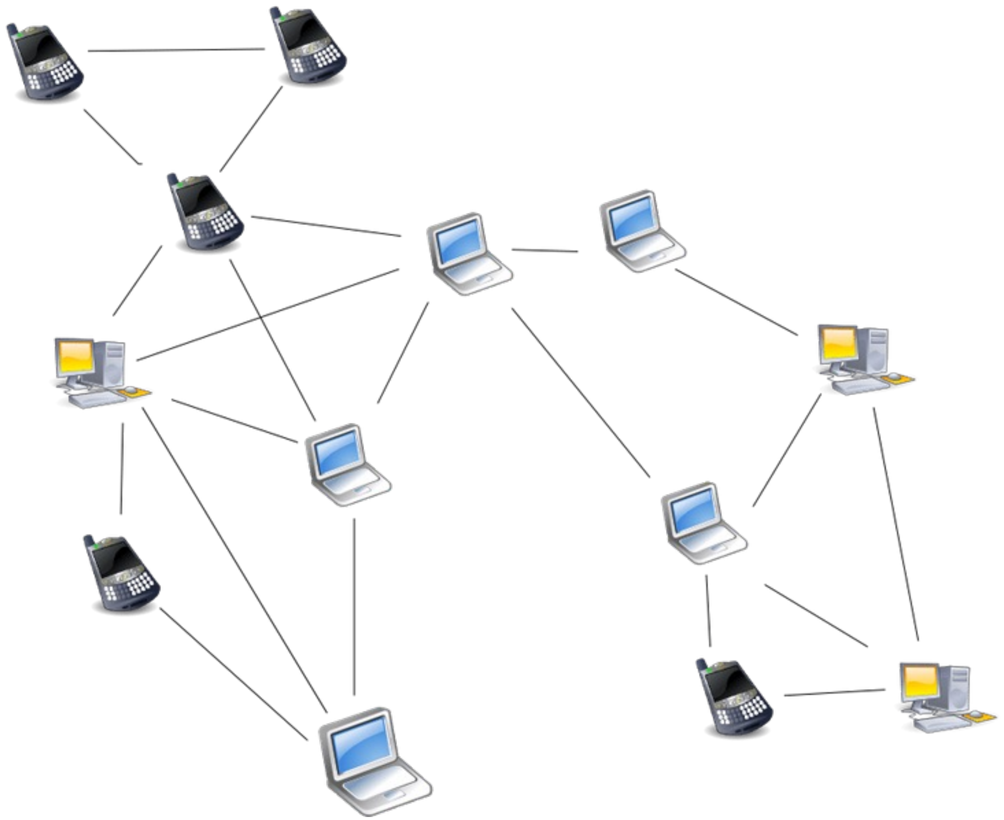
High resiliency to churn

- Nodes leave and join frequently

Nodes and resources are loosely-coupled

- Data navigation issue
- High resource (CPU, memory, network) usage

<u>Example:</u> Napster, Gnutella, KaZaA



Structured P2P network

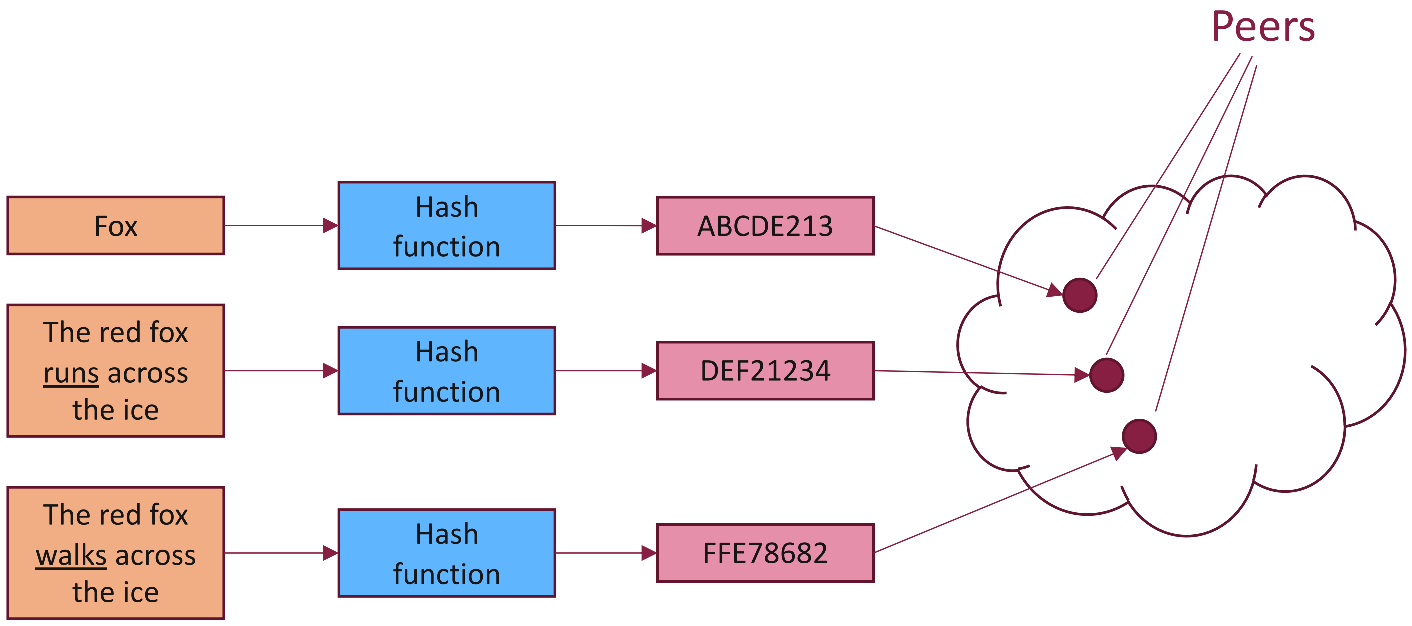
Structured overlay network, restriction on content placement and resource distribution

Nodes and resources are **tightly-coupled**, everyone has their own task

- Each node is responsible for a specific role in the network
- Distributed Hash Table (DHT) for node-task assignment
- Simplifying content location

Harder to build

Low resiliency against churn



Hybrid P2P network

Central authorities to help nodes navigate each other

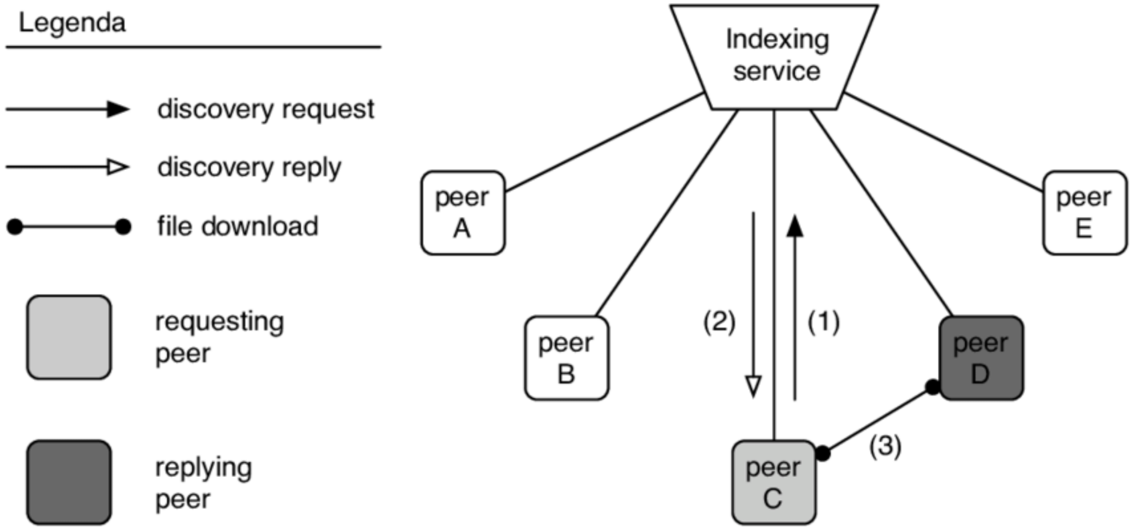
- Combine client-server with P2P models

Tend to improve overall performance

- Trade-off b/w centralization vs. node equality

Inherit the best of both worlds

- Efficiency in C-S setting, and decentralization in P2P setting



Consensus Mechanism

Consensus Mechanism

Main Motivation: Reliability and Fault-Tolerance in distributed system

- Correct operation in the presence of corrupted nodes
- Reach a common agreement in a distributed/ decentralized system
- Nodes propose values
  - All nodes must agree on one of these values

Consensus Mechanism

Key to solving many problems in distributed computing

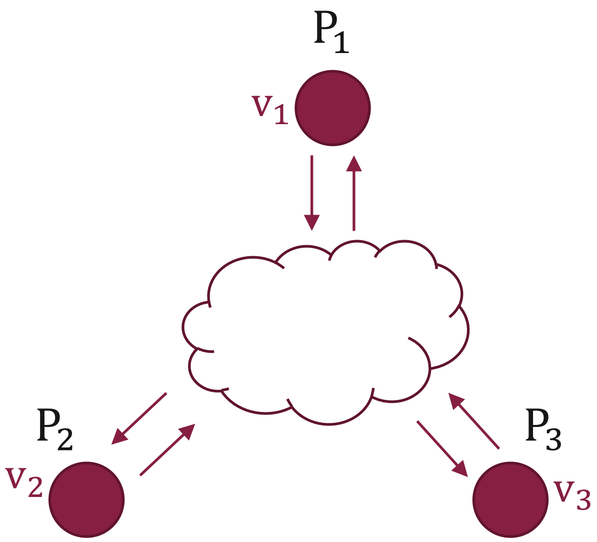
- Atomic broadcast
- Atomic commit of database transaction
- Clock synchronization
- Dynamic group membership

Consensus Protocol: Definition

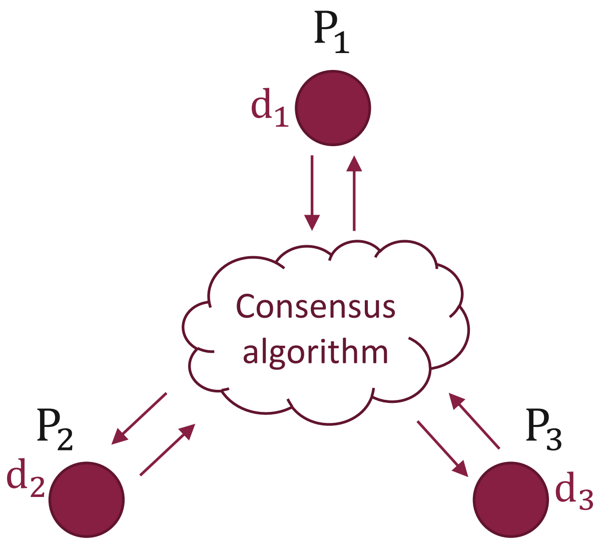
A consensus protocol comprises two algorithms:

- $v_i \leftarrow \mathrm{Propose}()$: Each node n_i propose a value v_i and broadcast v_i to the network
- $v \leftarrow \mathrm{Decide}(v_1, \dots, v_n)$: All nodes agree a common value $v \in \{v_1, \dots, v_n\}$

- The protocol terminates when all correct nodes decide on the same value
- The agreed value cannot be arbitrary: it must come from some correct node

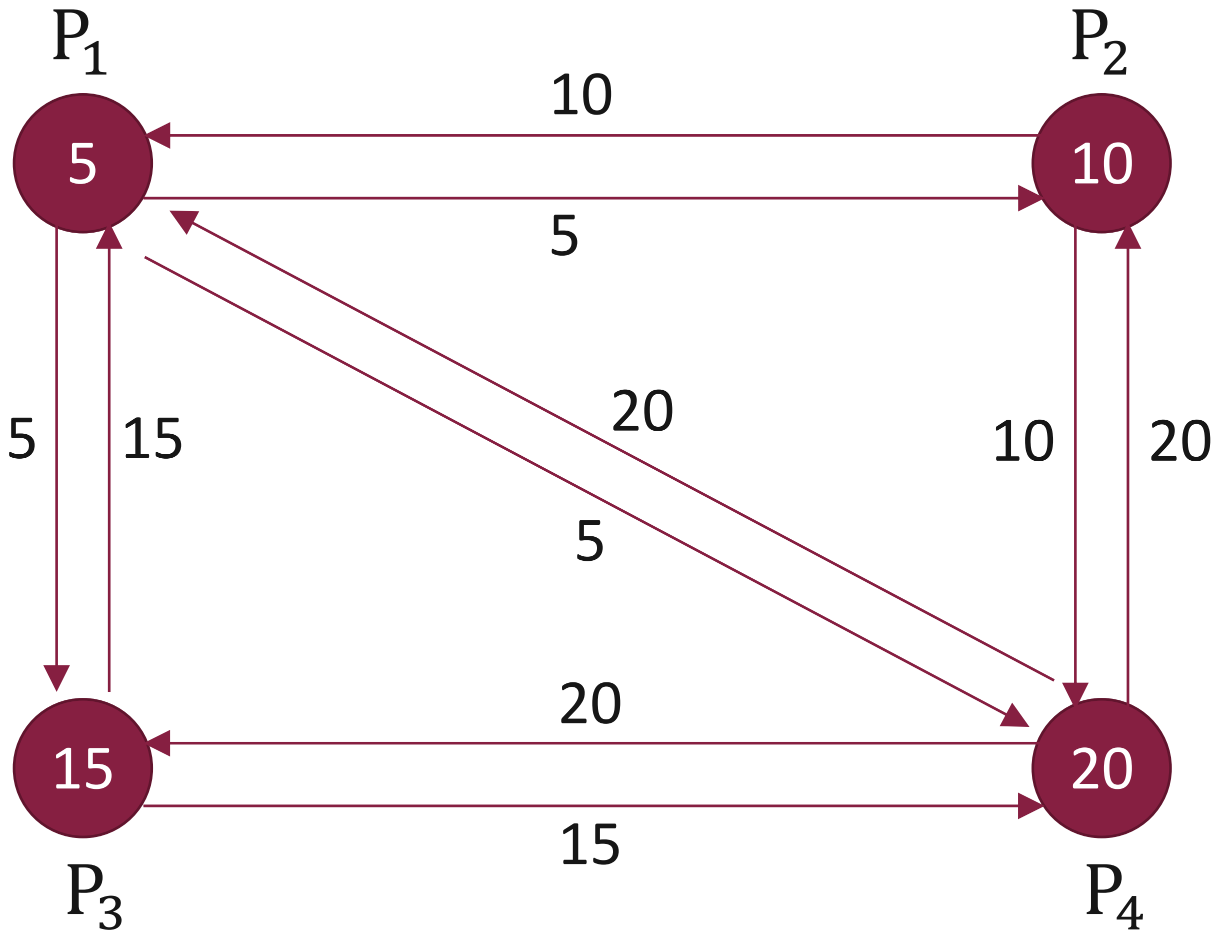


1. Propose



2. Decide

Example: Find max value among all values



Consensus Properties

Validity

Value agreed is a value proposed

Agreement

All correct nodes agree on the same value

Integrity

Every correct node decides at most once

Consensus Properties

Termination

Every correct node must decide at the end of protocol

Safety

Every correct node must not agree on incorrect value

Liveness

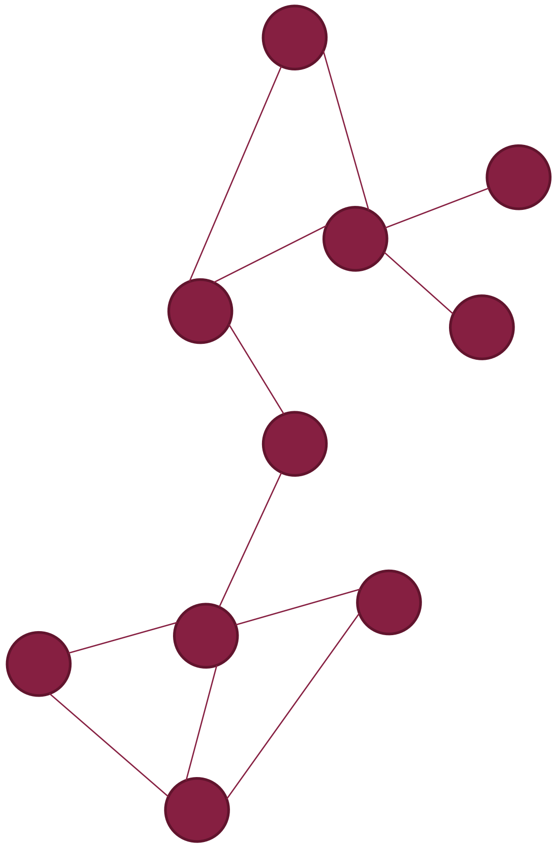
Every correct value must be accepted

When Failure Happens

If no failure or malice, easy to reach a consensus

- Individuals broadcast their values to all nodes
- Values received with a pre-defined timeframe (synchronous)

What if there are failures or malicious activities in the network?



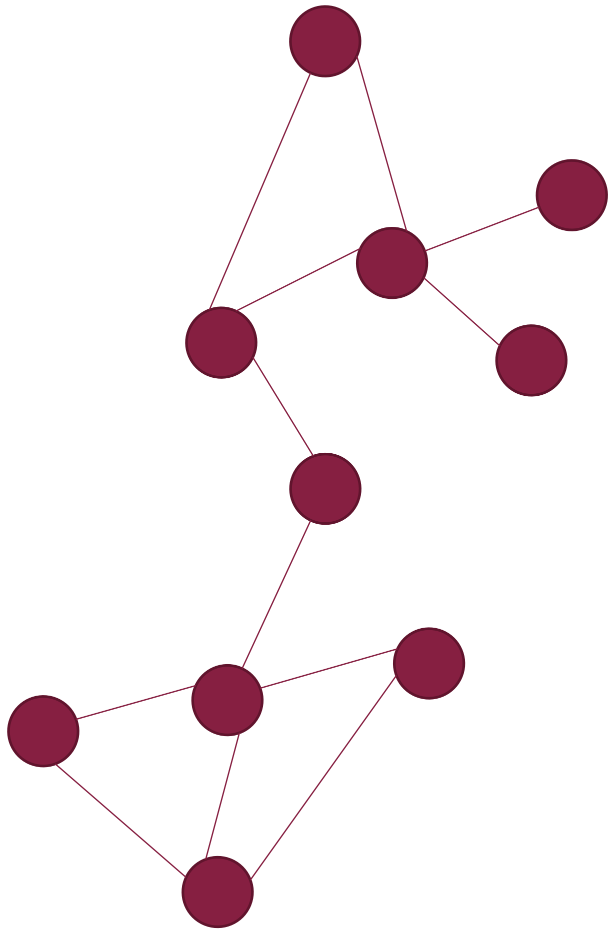
When Failure Happens

Common types of failure

Crash Fault: Node crashed, offline during communication

Network Fault: Not all pairs of nodes well-connected (partitioned network), latency (no notion of global time)

Byzantine Fault: Nodes may be malicious



Achieving consensus in the faulty (yet realistic) environment is hard

Synchronous vs. Asynchronous Systems

Synchronous system

- Defined maximum waiting time for message transmission
- Easy to reach a consensus

Asynchronous system

- Undefined waiting time
- Hard to achieve a consensus

Impossibility Results

(Another) Impossibility Results

Byzantine Generals Problem

“Consensus is impossible with a single faulty node”

- Fischer, Lynch and Patterson

Choose 2 out of 3: Safety, Liveness, Fault-Tolerance

Impossibility Results

Understand IRs correctly

- IRs are more about the model than about the problem
- Developed to study systems like distributed databases

Blockchain has different model

- Consensus is still useful and achievable
- Find right algorithm for specific application domain

Consensus Algorithms

Paxos

Majority rule, asynchronous setting

Consistency, fault-tolerance, but may get stuck (2 out of 3 rule)

Byzantine-fault intolerance

Raft

Leader-Follower model

Choose 2 in 3: Safety, Liveness, Fault-Tolerance

Byzantine-fault intolerance

http://thesecretlivesofdata.com/raft/ (animated example)

BFT

Byzantine-fault tolerance

Stay tuned for next lecturers!

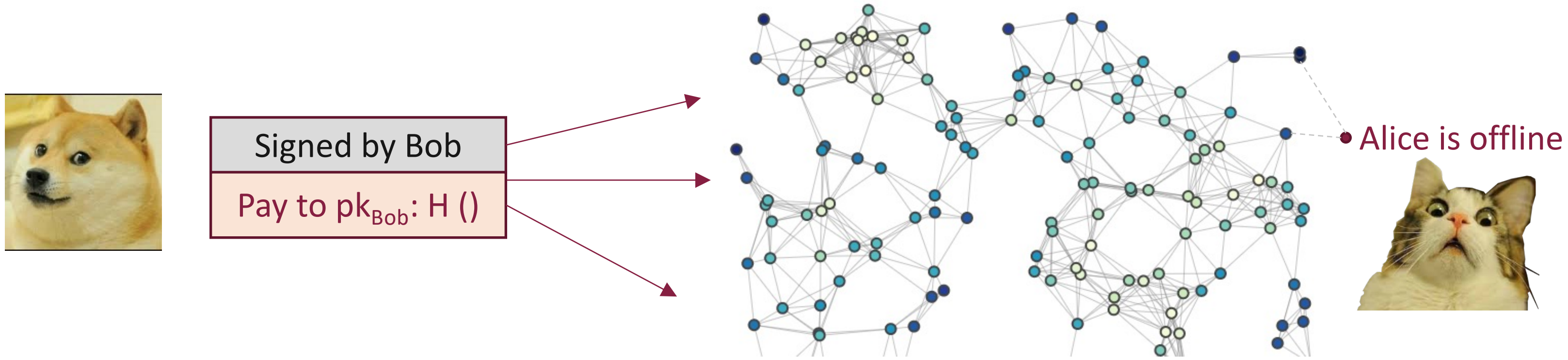
Consensus in Public Blockchain

Traditional consensus works on closed environment

- Nodes know addresses of their peers
- Every node accesses a shared memory

Public blockchain is an open P2P system

- Where to keep shared memory in P2P?
- Anyone can join and leave the network at anytime
- How to enable consensus in an open system?



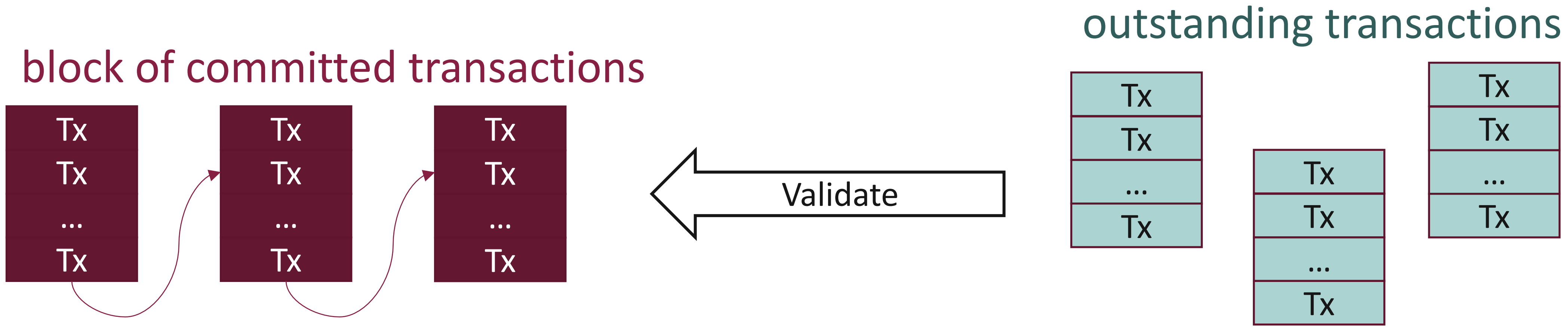
All nodes must agree on the validity of the Bob's transaction

Consensus in Public Blockchain

At any given time:

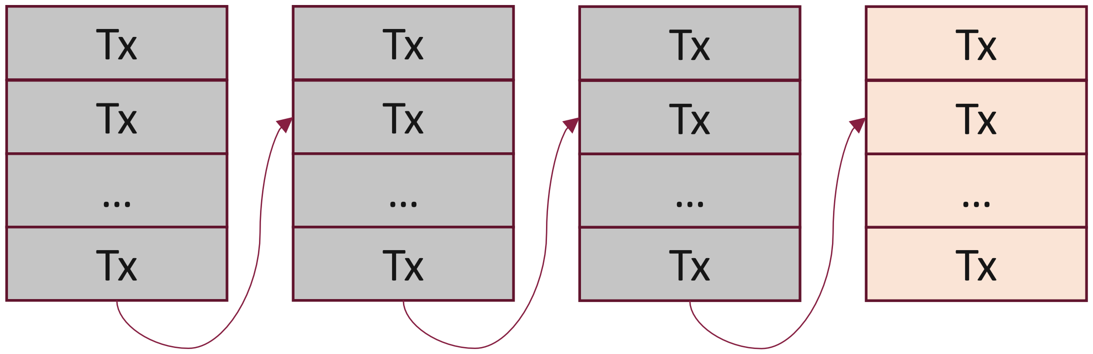
All nodes have a sequence of blocks of transactions
they have reached a consensus on (block of committed transactions)

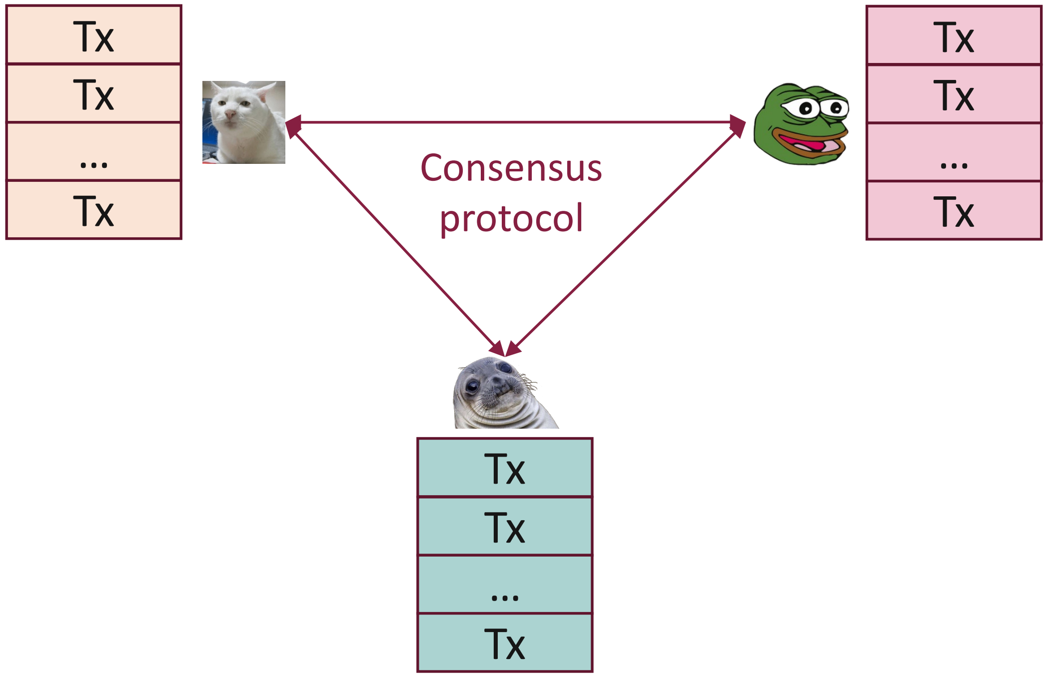
Each node has a set of outstanding transactions
that need to be validated against block of committed transactions



Block-based consensus

Blockchain Consensus





Consensus in Public Blockchain

Bitcoin introduces incentive concept for honest actions

- Possible as Bitcoin is a digital currency

Embrace randomness

- Does away with the notion of a specific end-point
- Consensus happens over long-time scales – approx. 1 hour

Blockchain consensus works better in practice than in theory

- Theory is catching up
- Theory is still very important as It can help predict unforeseen attacks

We will find out in more details later…