Foundational Layer(s) for Blockchain

Blockchain Engineering 2024

Bulat Nasrulin

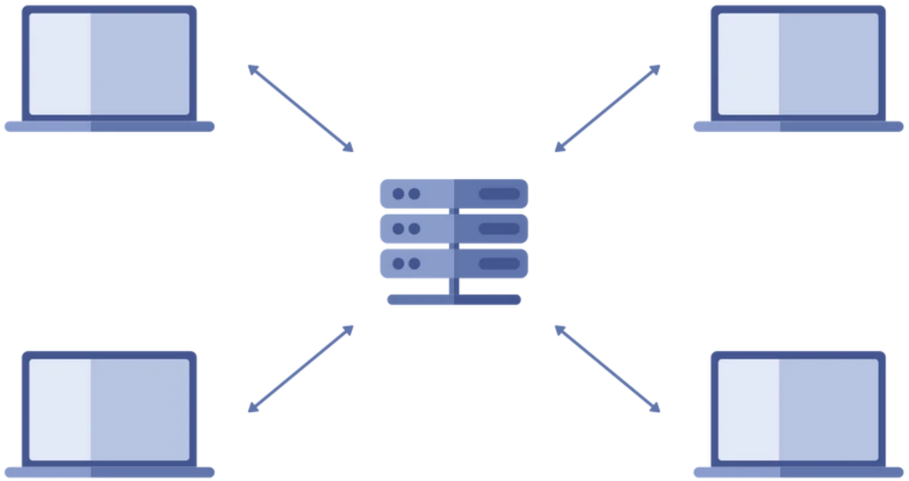
TU Delft

This Lecture

- Introduction to Peer-to-Peer technology
- Giving you the right mindset for your projects
- How it is done in ipv8

- First Deep Dive into typical Blockchain network
- Demo for Toy Blockchain on py-ipv8

Network Architectures



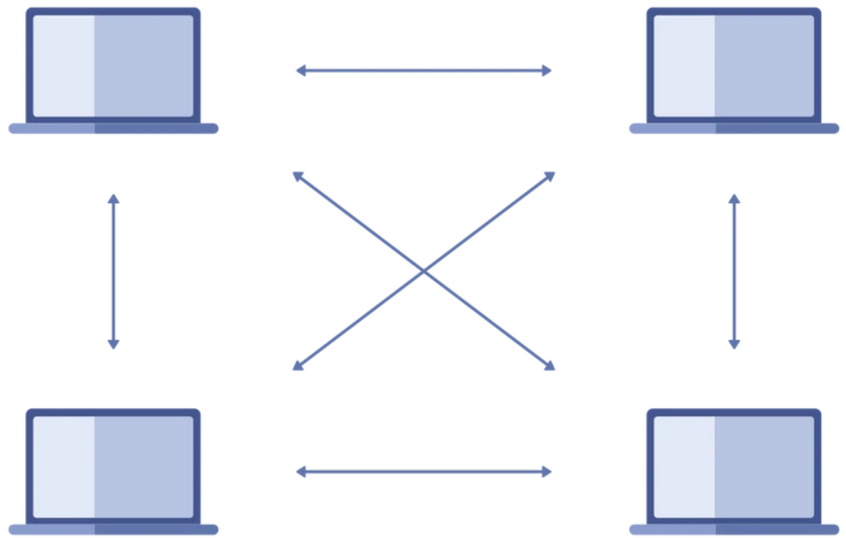
Centralized network

- Typical client-server architecture
- TCP connection (Reliable, Ordered)
- Centralized data management

TUDelft

Network Architectures

- Peers: Server-to-Server
- No reliable connections
- Anything global or consistent is challenging (sometimes impossible)
- Data management is super hard: concurrency and reliability issues



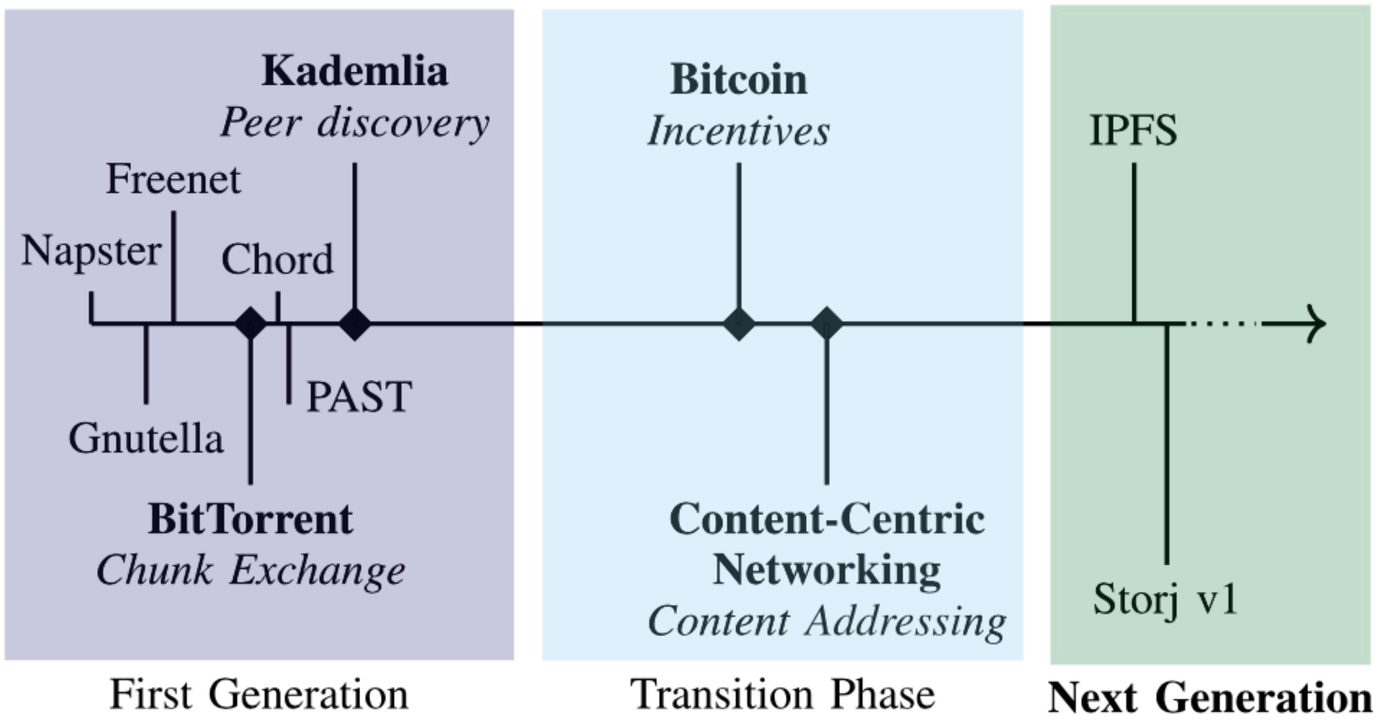
Peer-to-Peer network

Our Dream: We can achieve more through collaboration and shared resources

Our Reality: A lot of problems to solve

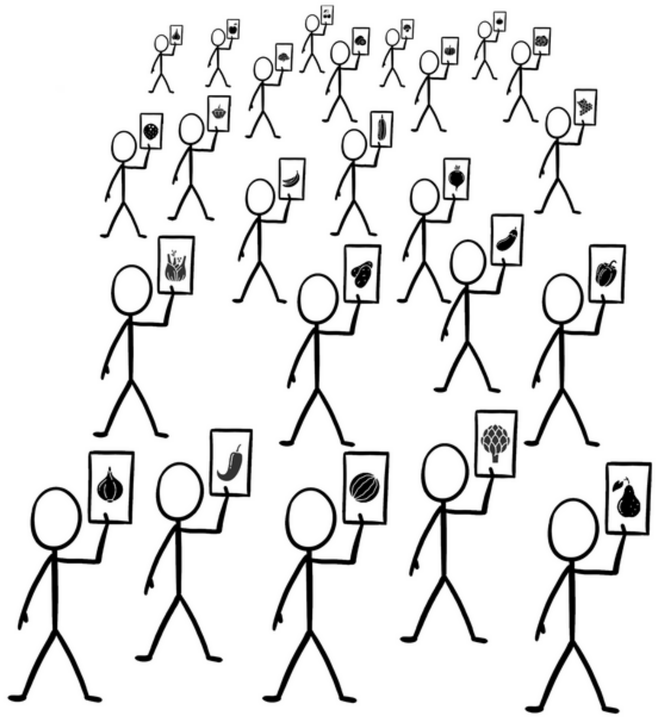
TU Delft

P2P Evolution



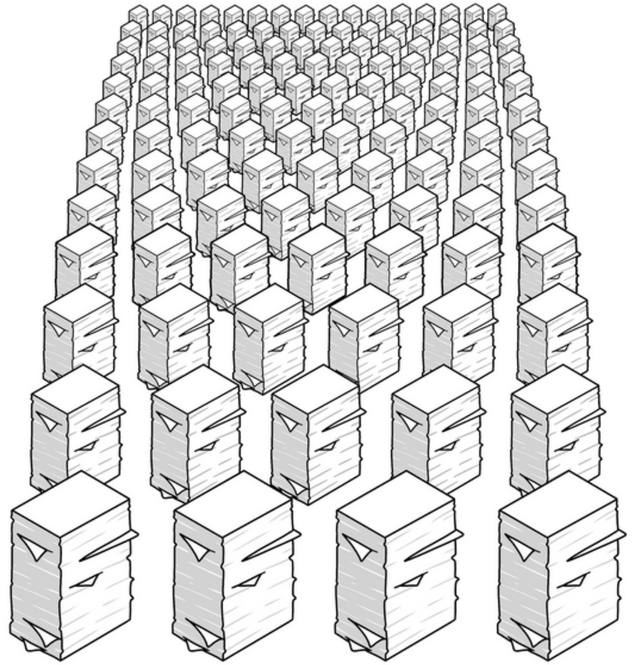
Rich History and many proposals

Global vs Local State



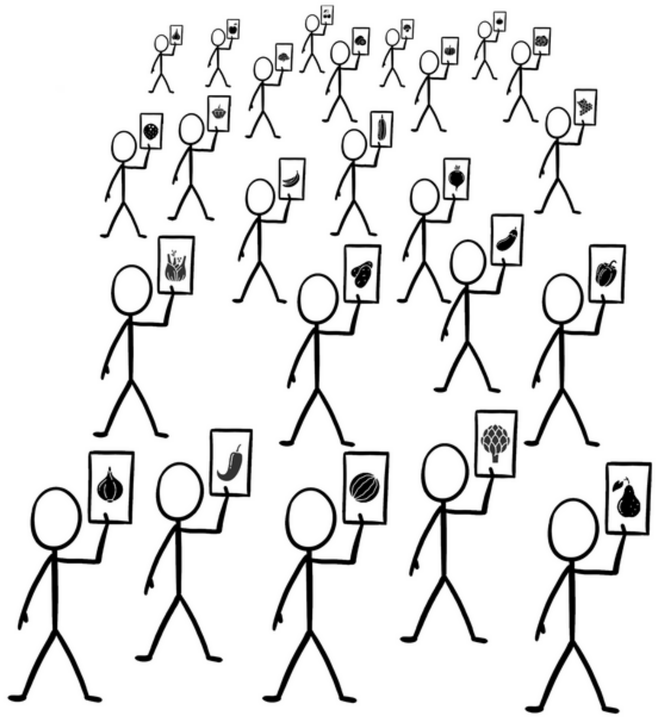
Local State

Global Consensus Protocol



Global State

Global vs Local State

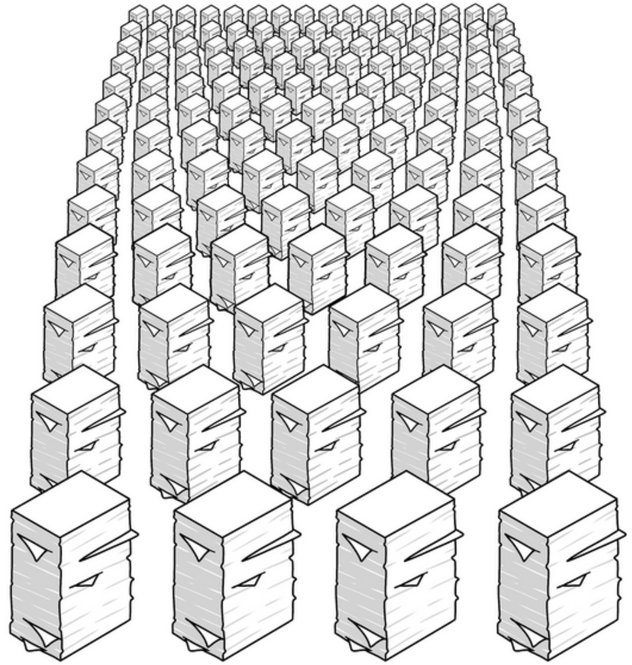


Local State

default, local database state

Global Consensus Protocol

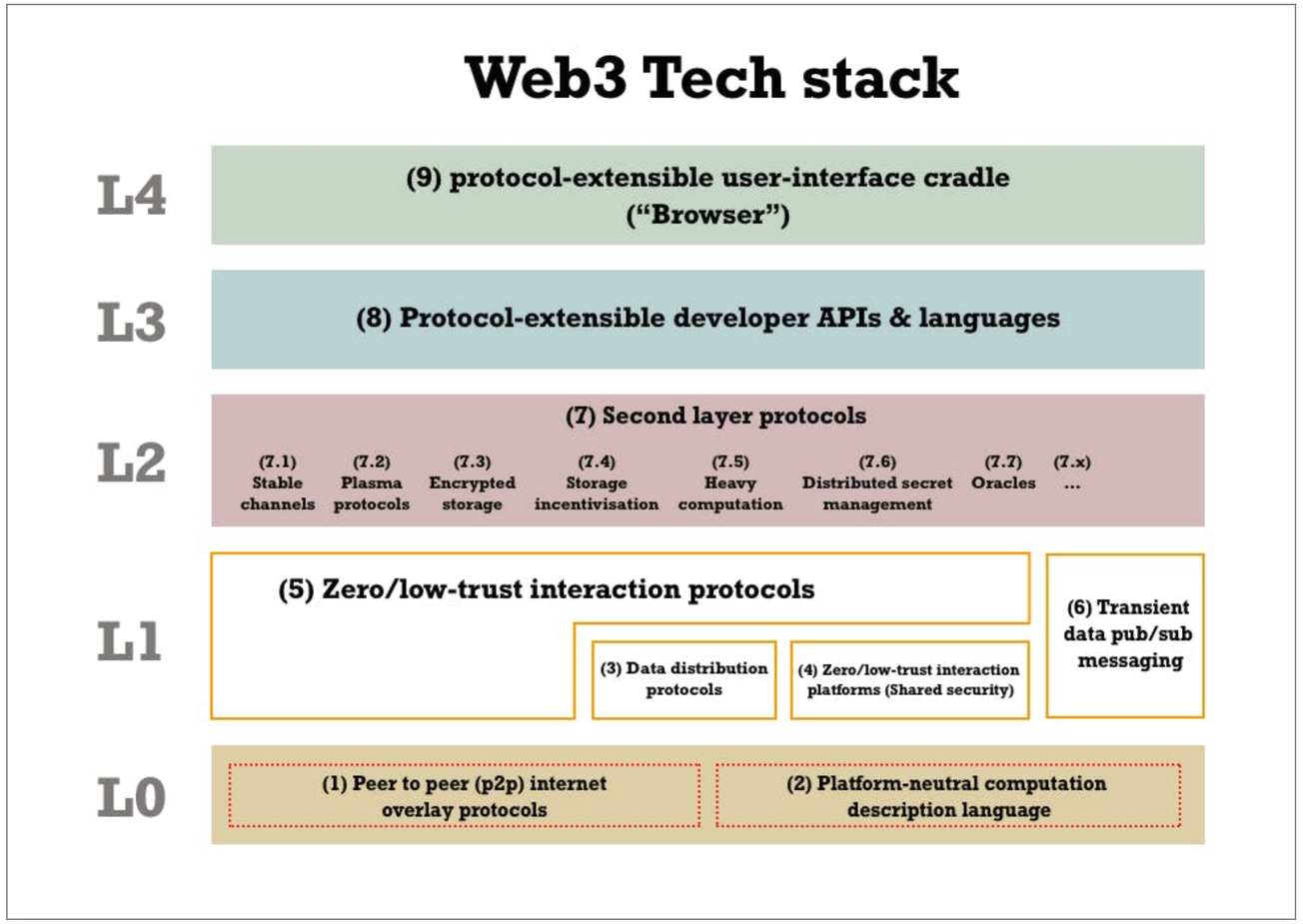
Will be covered in the next lectures



Global State

Shared replicated state

TUDelft



Layer 2

Scaling: channels, sidechains, rollups

Layer 1

Execution protocols

Data Consistency (Consensus) Protocols

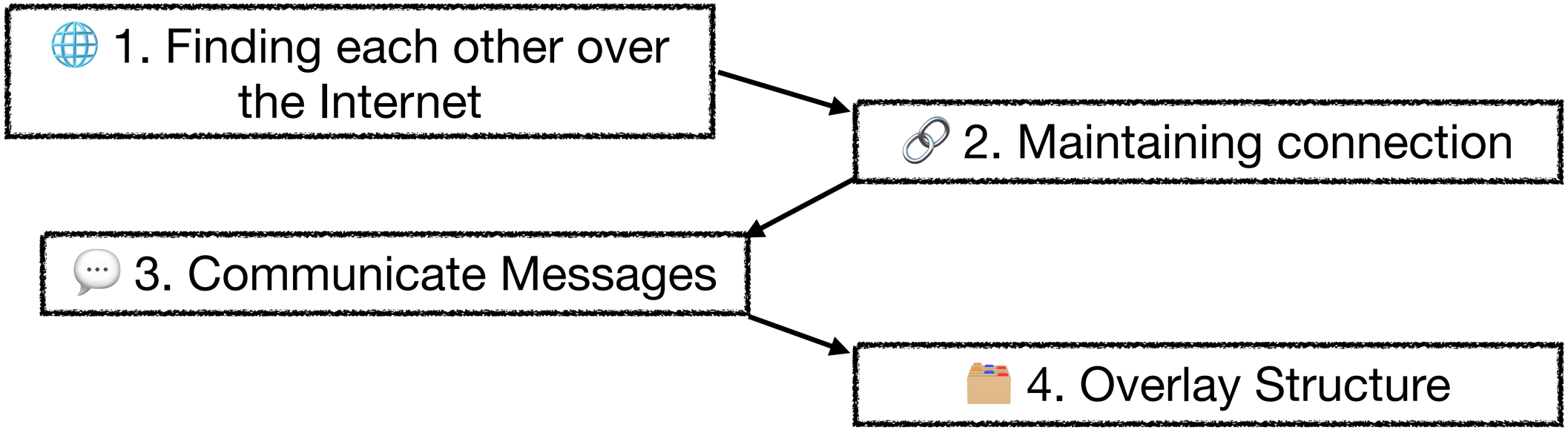
Layer 0

Data Delivery (Gossip) Protocols

P2P Overlay Protocols

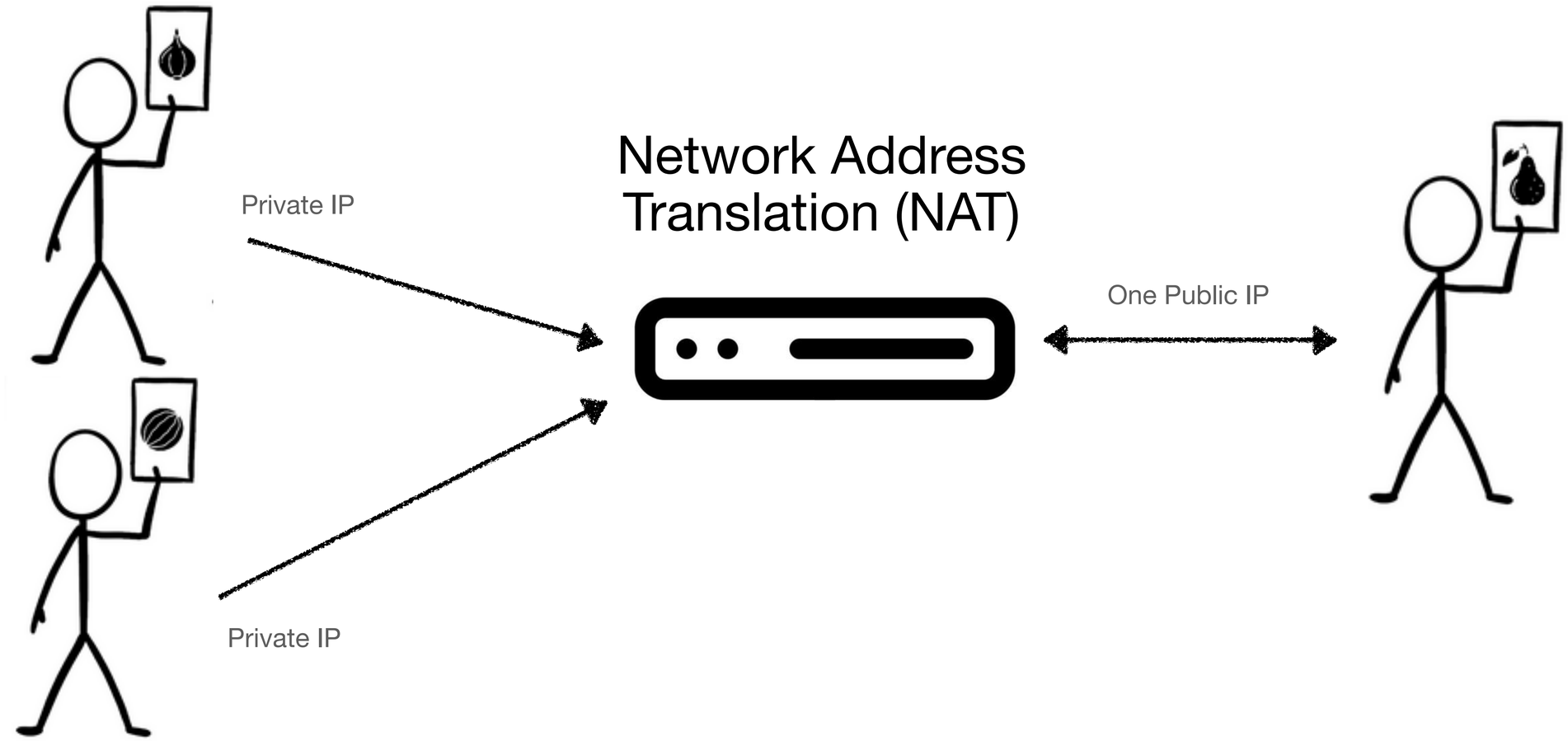
TUDelft

Before Consensus

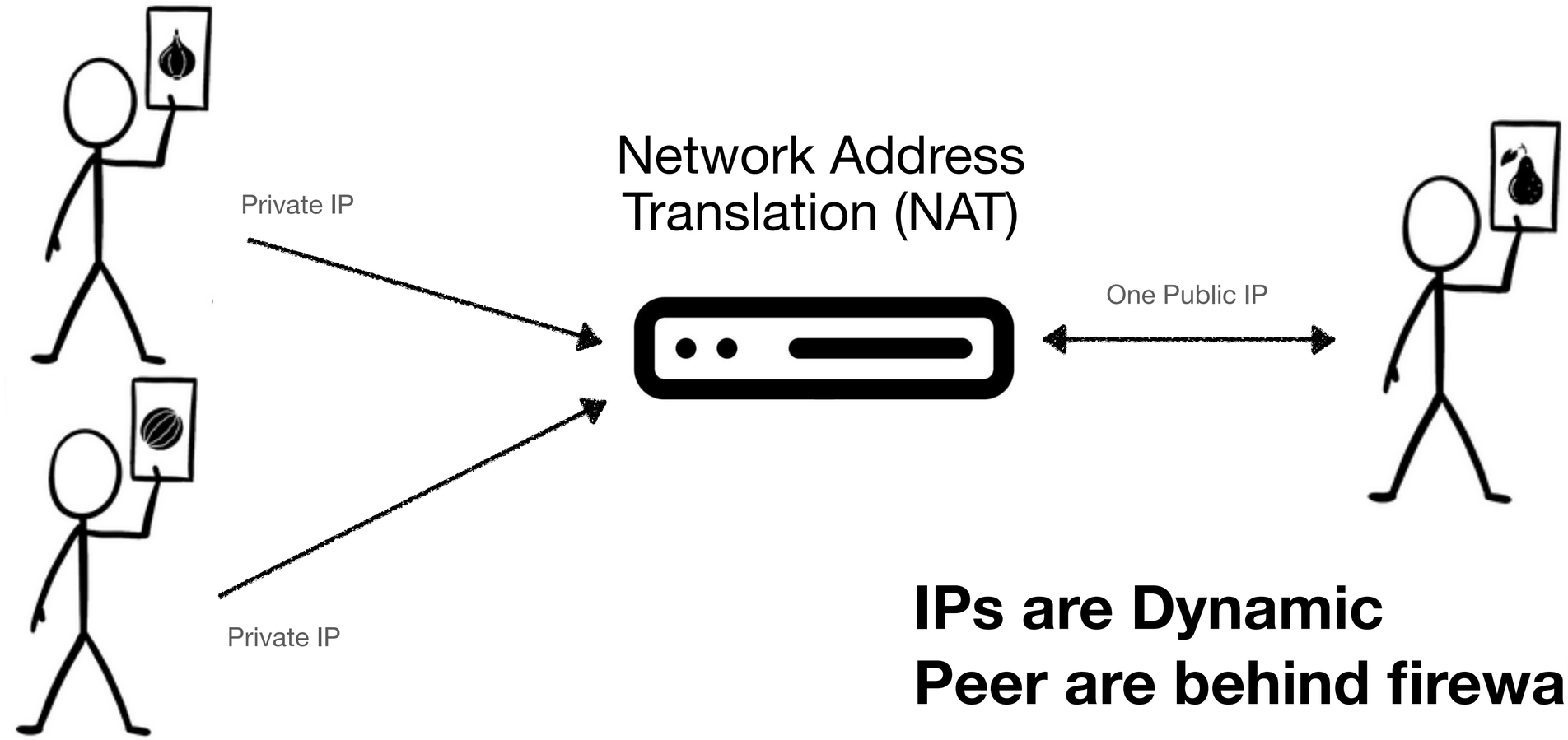


🌐 1. Finding each other over the Internet

🌐 Finding Each Other

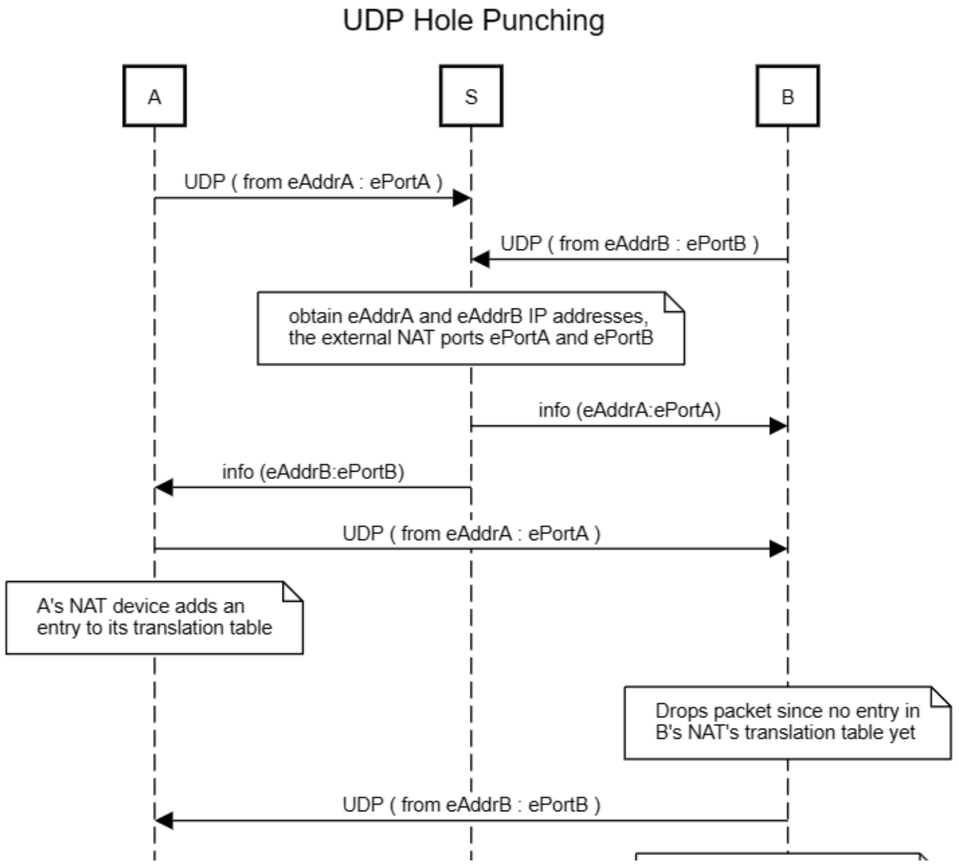


🌐 Finding Each Other



IPs are Dynamic
Peer are behind firewalls

🌐 Finding Each Other: NAT Hole punching

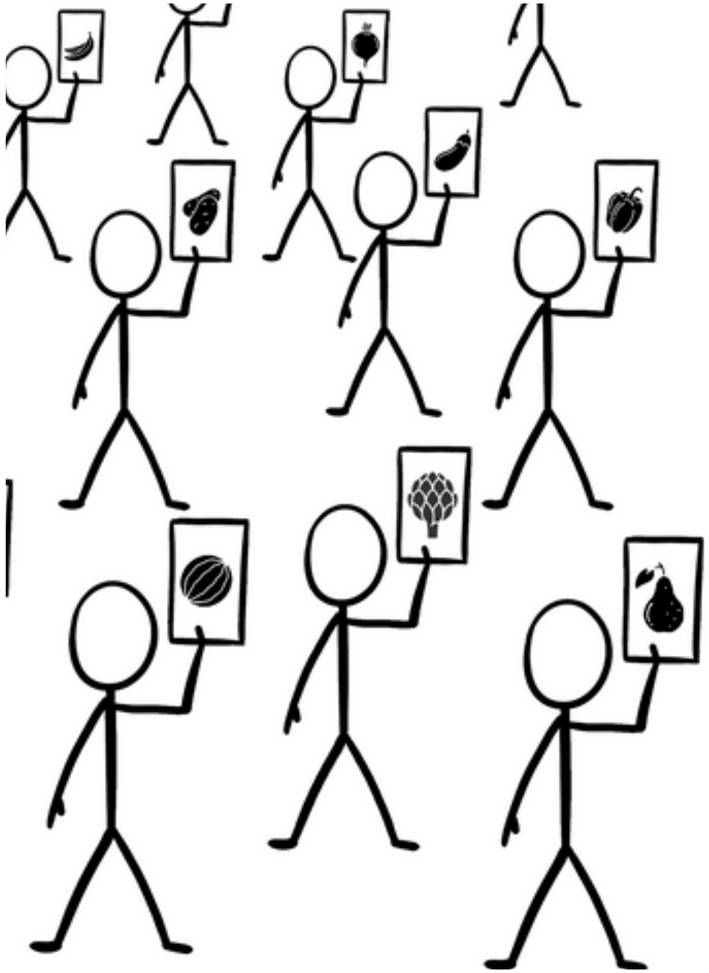


UPnP (Universal Plug and Play): Automatic port forwarding.

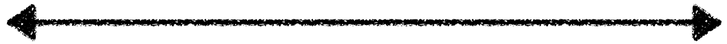
NAT Hole Punching: Coordinated connection attempts to establish a direct pathway.

STUN/TURN servers: External servers to facilitate connection establishment.

🌐 Finding Each Other: Bootstrap



How do I find who is online?

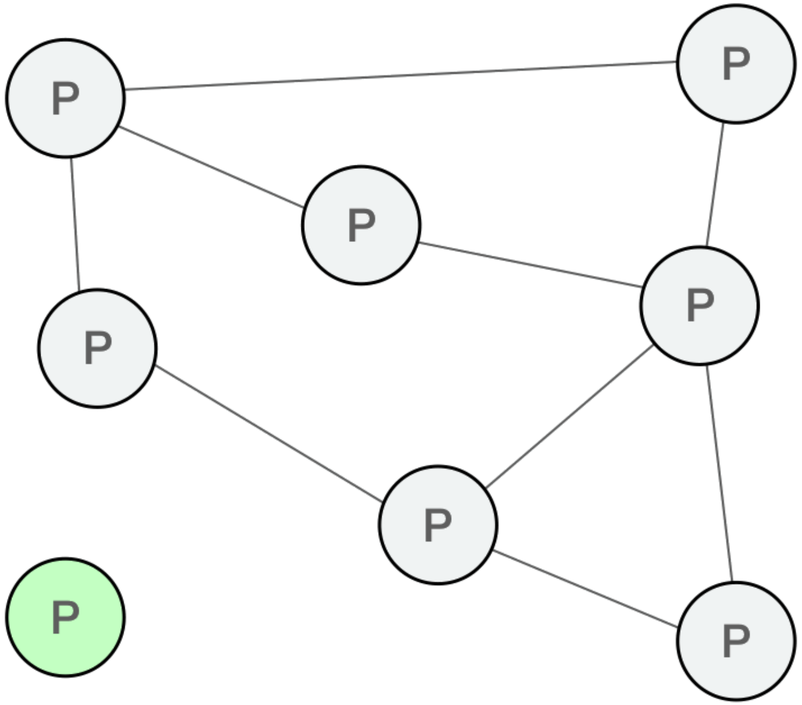


🌐 Finding Each Other: Bootstrap

Using hardcoded bootstrap servers

TU Delft

🌐 Finding Each Other: Bootstrap



Connect to Bootstrap Servers

Handshake Introduction

Bootstrap Server gives back list of online peers

```
// From: https://github.com/bitcoin/bitcoin/blob/master/src/chainparams.cpp

vSeeds.emplace_back("seed.bitcoin.sipa.be"); // Pieter Wuille, only supports
vSeeds.emplace_back("dnsseed.bluematt.me"); // Matt Corallo, only supports x
vSeeds.emplace_back("dnsseed.bitcoin.dashjr.org"); // Luke Dashjr
vSeeds.emplace_back("seed.bitcoinstats.com"); // Christian Decker, supports
vSeeds.emplace_back("seed.bitcoin.jonasschnelli.ch"); // Jonas Schnelli, onl
vSeeds.emplace_back("seed.btc.petertodd.org"); // Peter Todd, only supports
vSeeds.emplace_back("seed.bitcoin.sprovoost.nl"); // Sjors Provoost
vSeeds.emplace_back("dnsseed.emzy.de"); // Stephan Oeste
```

Hardcoded DNS servers for Bootstrap

https://developer.bitcoin.org/devguide/p2p_network.html#:

Bitcoin Network

REACHABLE BITCOIN NODES
Updated: Tue Feb 20 14:50:07 2024 CET

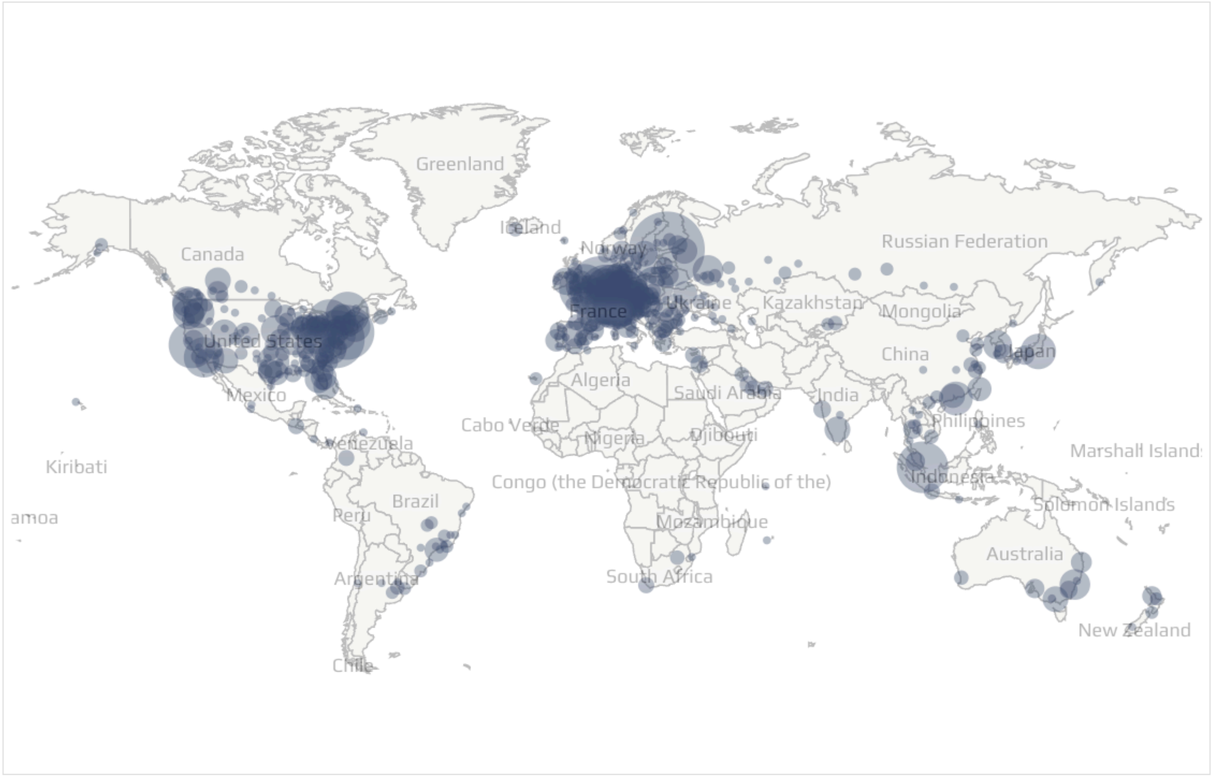
18077 NODES **CHARTS**

IPv4: +4.4% / IPv6: +17.0% / .onion: +7.0%

Top 10 countries with their respective number of reachable nodes are as follows.

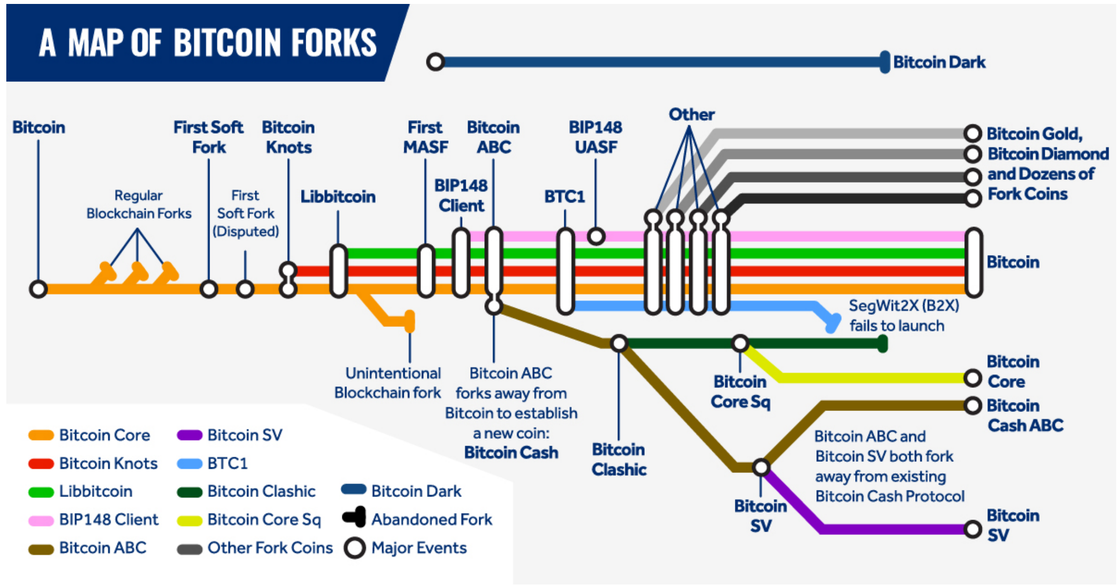
RANK	COUNTRY	NODES
1	n/a	11394 (63.03%)
2	Germany	1642 (9.08%)
3	United States	1585 (8.77%)
4	France	411 (2.27%)
5	Netherlands	340 (1.88%)
6	Finland	298 (1.65%)
7	Canada	281 (1.55%)
8	United Kingdom	201 (1.11%)
9	Russian Federation	156 (0.86%)
10	Singapore	148 (0.82%)

All (95) »



Map shows concentration of reachable Bitcoin nodes found in countries around the world. **LIVE MAP**

Bitcoin Connection and Bootstrap



1. Exchange versions. Connect if compatible

2. Peers keep a list of active peers

Known active peers

> The typical presumption is that a node is likely to be active if it has been sending a message within the last three hours.

3. Request ‘getaddr’ list of active peers from other

TUDelft

Default Policies

Bitcoin Core: at least 8 outgoing connections and at most 125

IPv8 Default: min_peers = 20, max_peers = 30

TUDelft

Default Policies

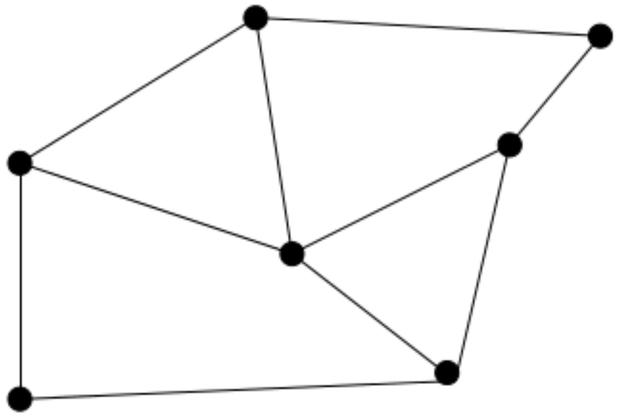
Bitcoin Core: at least 8 outgoing connections and at most 125

IPv8 Default: min_peers = 20, max_peers = 30

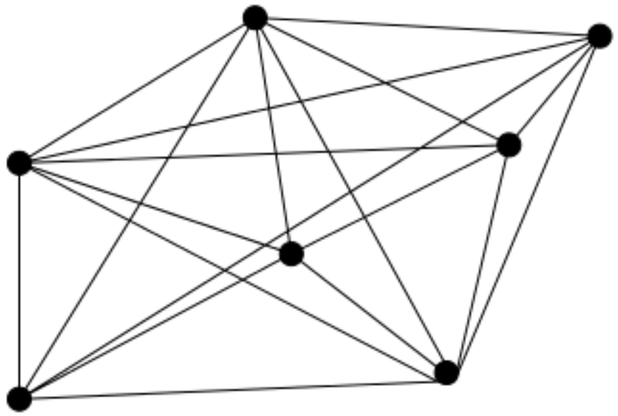
What is the effect of this value?

TUDelft

Default Policies



sparse

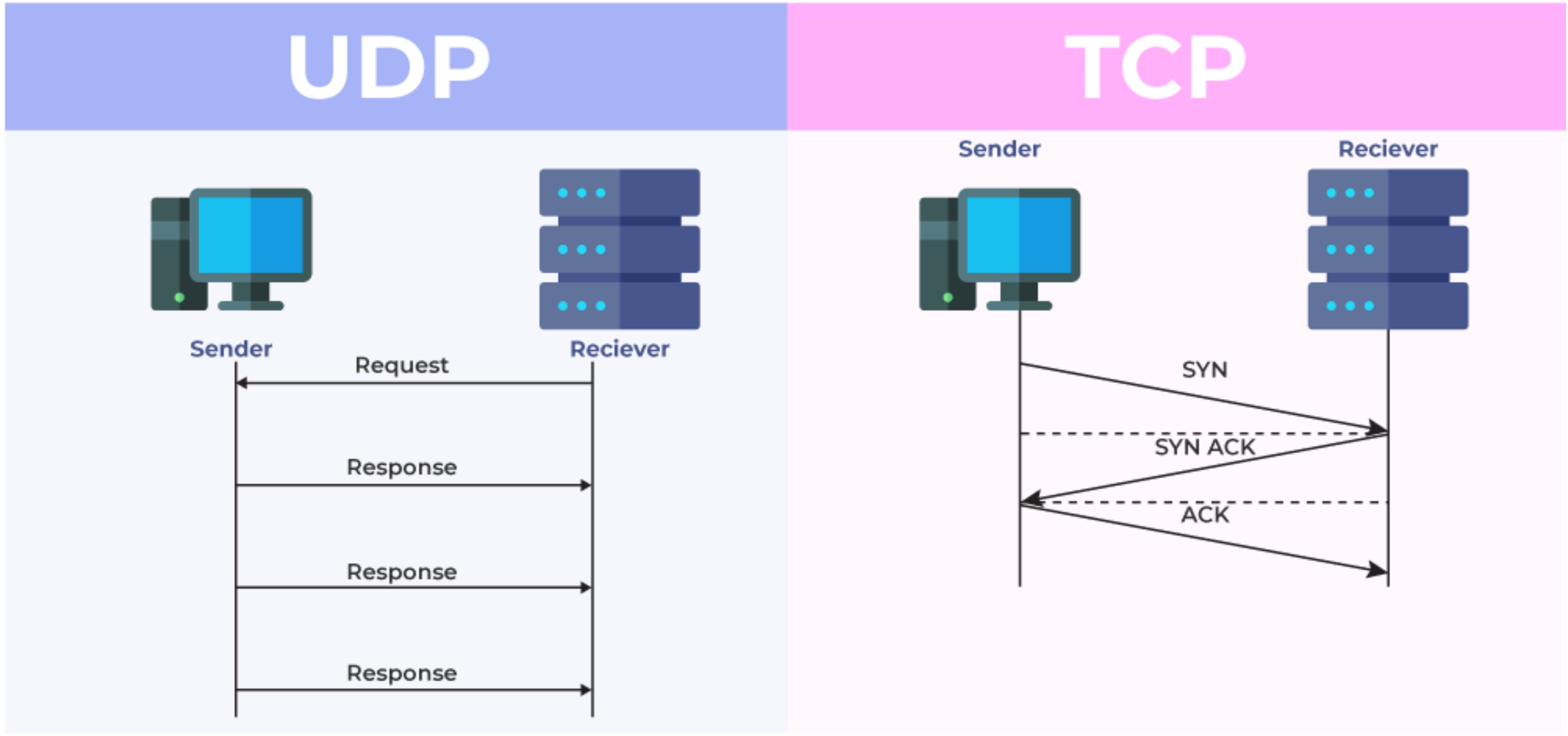


dense

What is the effect of this value?

🔗 2. Maintaining connection

Communication Protocol



Which one is better for P2P?

Communication Protocol

UDP	TCP
Lower Latency	More reliable
Better with churn	Has retransmission
Max packet size: 65507 bytes (lower in practice)	Fragmentation for bigger blobs
IPv8 choice	Bitcoin choice

Communication Protocol

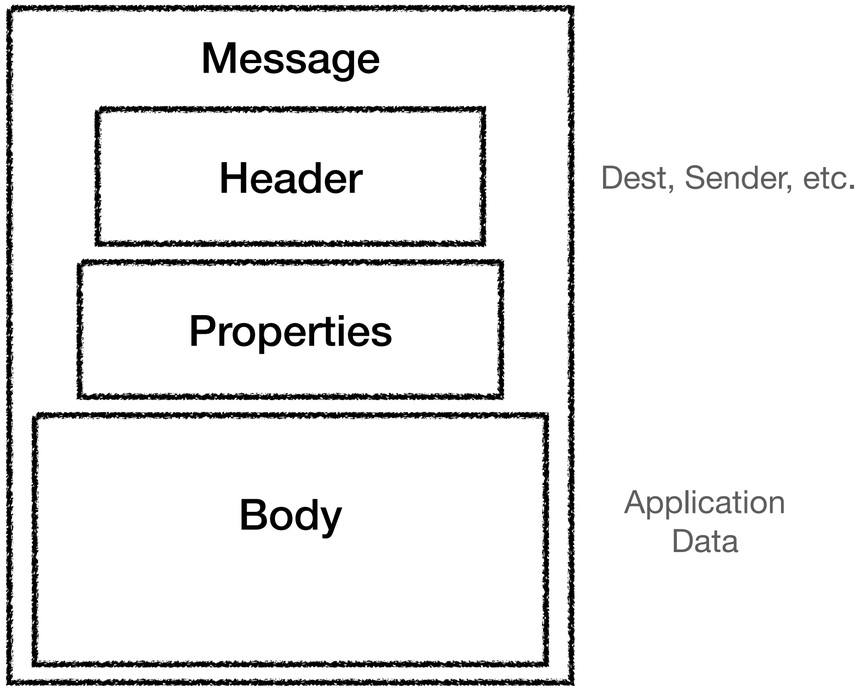
Bitcoin: nodes by default will send a message to peers before **30 minutes of inactivity**. If 90 minutes pass without a message being received by a peer, the client will assume that connection has closed

IPv8: Ping request after **30 seconds of inactivity**. If a peer does not respond to a ping request, it will be removed from the Network. By default the strategy checks eight peers simultaneously. A total three pings are sent, one every ten seconds, after 30 seconds of inactivity. These settings should be adjusted if a large number of peers are being connected to.

TUDelft

💬 3. Communicate Messages

What is a Message



Serialized into bytes one way

When deserialized executes with some application logic

How do we distribute message

Push:

- When received first time forward message to F other nodes at most TTL times (ttl is inside the message)

Pull:

- Periodically contact the other node and ask for new messages

Epidemic Protocols

Push:

- When received first time forward message to F other nodes at most TTL times (ttl is inside the message)

Pull:

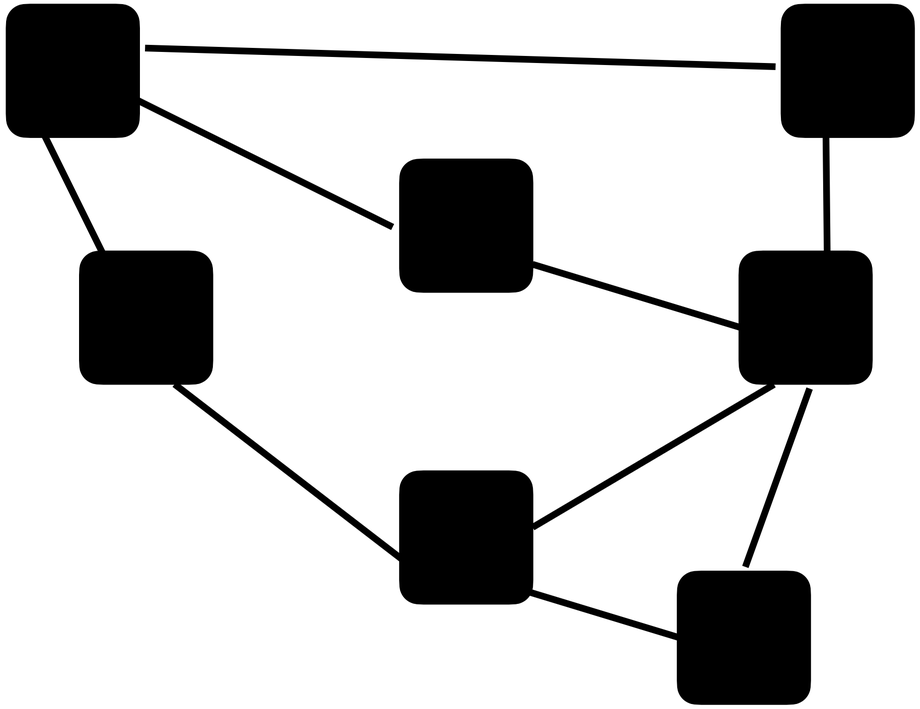
- Periodically contact the other node and ask for new messages

Which one is better?

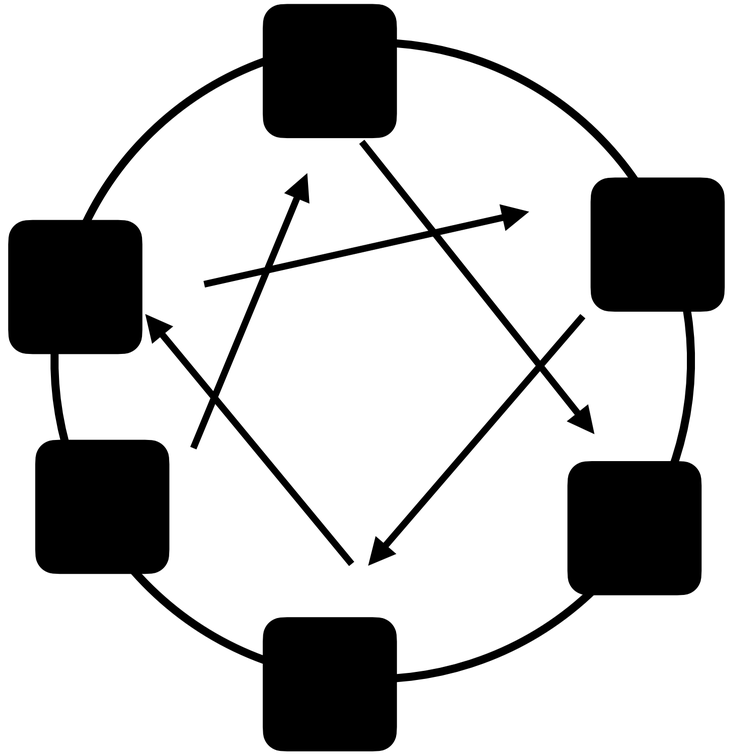
TUDelft

4. Overlay Structure

Two main ways to build overlays



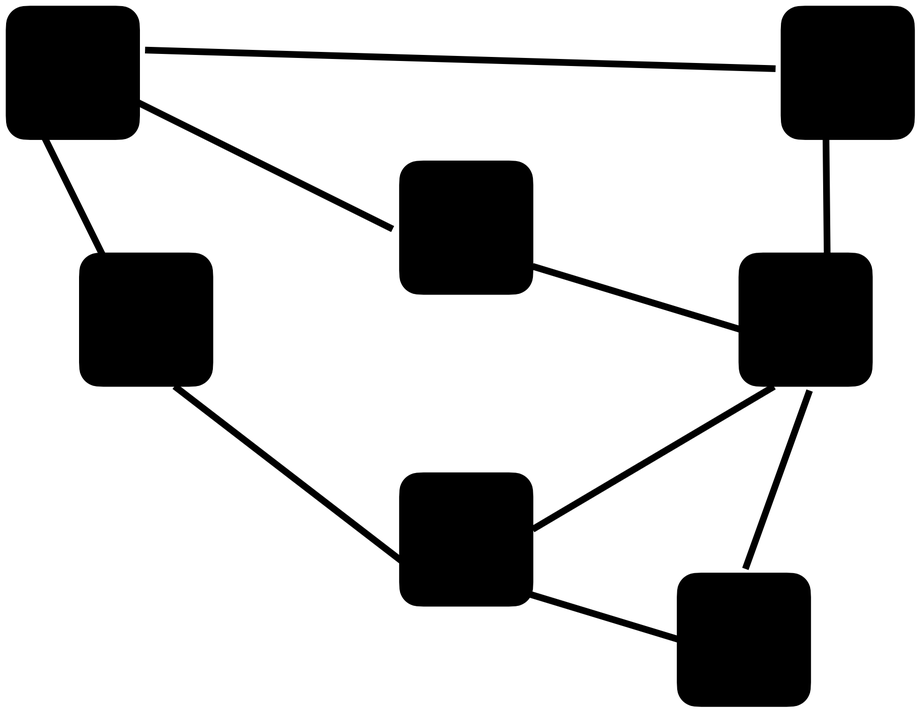
Structured overlay



Unstructured overlay

TUDelft

Two main ways to build overlays



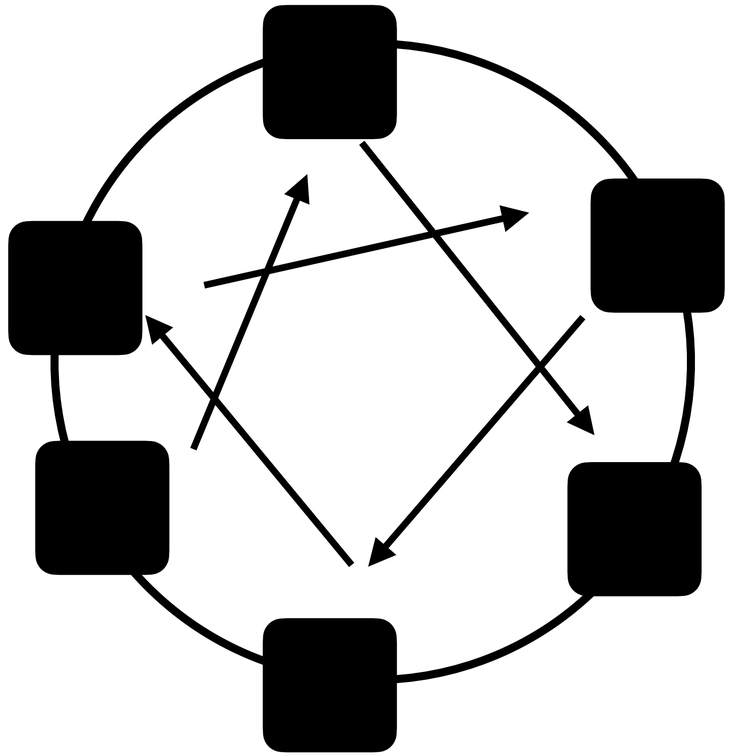
Unstructured overlay

- Each peer independently connects to random k peers
- Random network with ad-hoc protocols for search and storage
- Keep alive local peers use random gossip protocol

Which systems use it?

Two main ways to build overlays

- Use id of the peers to build structured overlay (like closest ids)
- Structured index, Distributed Hash Table (DHT)
- Rigid organizational principles for search, storage etc.



Unstructured overlay

Which systems use it?

Desirable Properties of P2P Collaboration

Open join and leave

Minimal overhead

Robust to Attacks, Failures

Scalable

Serendipity

Trustless

Permissonless

What is better for blockchain?

Structured overlay

Unstructured overlay

 Open join and leave?

What is better for blockchain?

Structured overlay

Unstructured overlay

Open join and leave?

Each leave or join require restructure

Peers can connect or disconnect with anyone

What is better for blockchain?

Structured overlay

Unstructured overlay

Open join and leave?

Minimal overhead?

What is better for blockchain?

Structured overlay

Unstructured overlay

Open join and leave?

Systematically Distributed

Minimal overhead?

Dynamically Shared

What is better for blockchain?

Structured overlay

Unstructured overlay

Open join and leave?

Minimal overhead?

Robust to Attacks,
Failures

What is better for blockchain?

Structured overlay

Unstructured overlay

Open join and leave?

Minimal overhead?

Relies on identity of the peers

Robust to Attacks, Failures

Multiple Redundant Paths

What is better for blockchain?

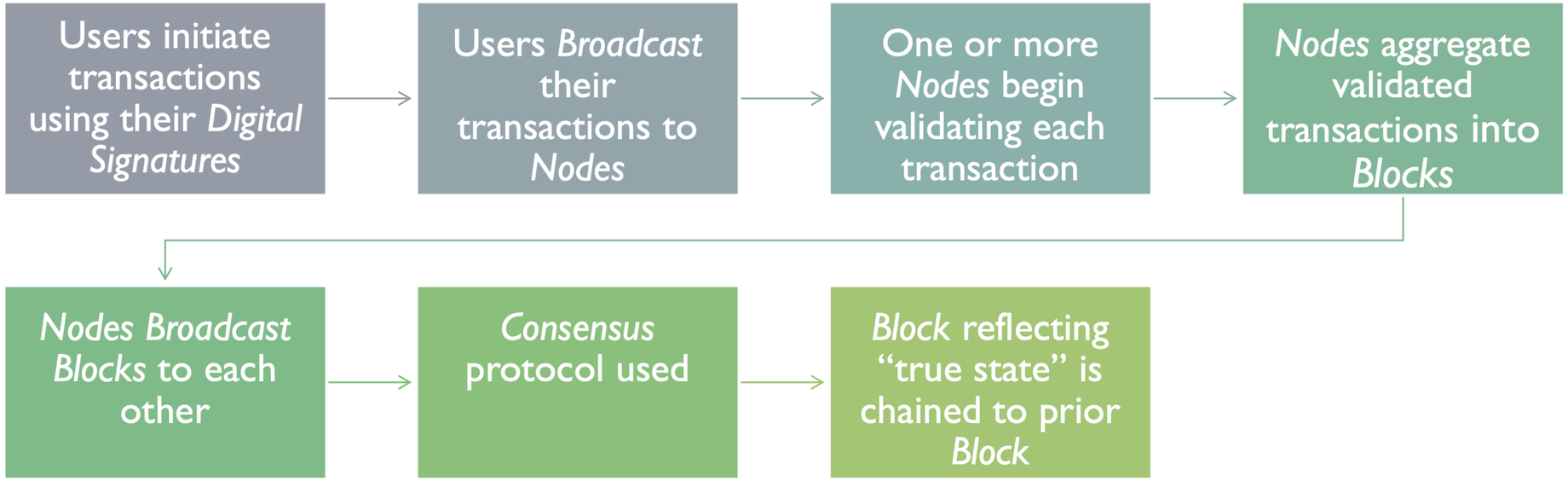
Structured overlay		Unstructured overlay
	👐 Open join and leave?	★
★	🪶 Minimal overhead?	
	🔒 Robust to Attacks, Failures	★
	📶 Scalable?	

What is better for blockchain?

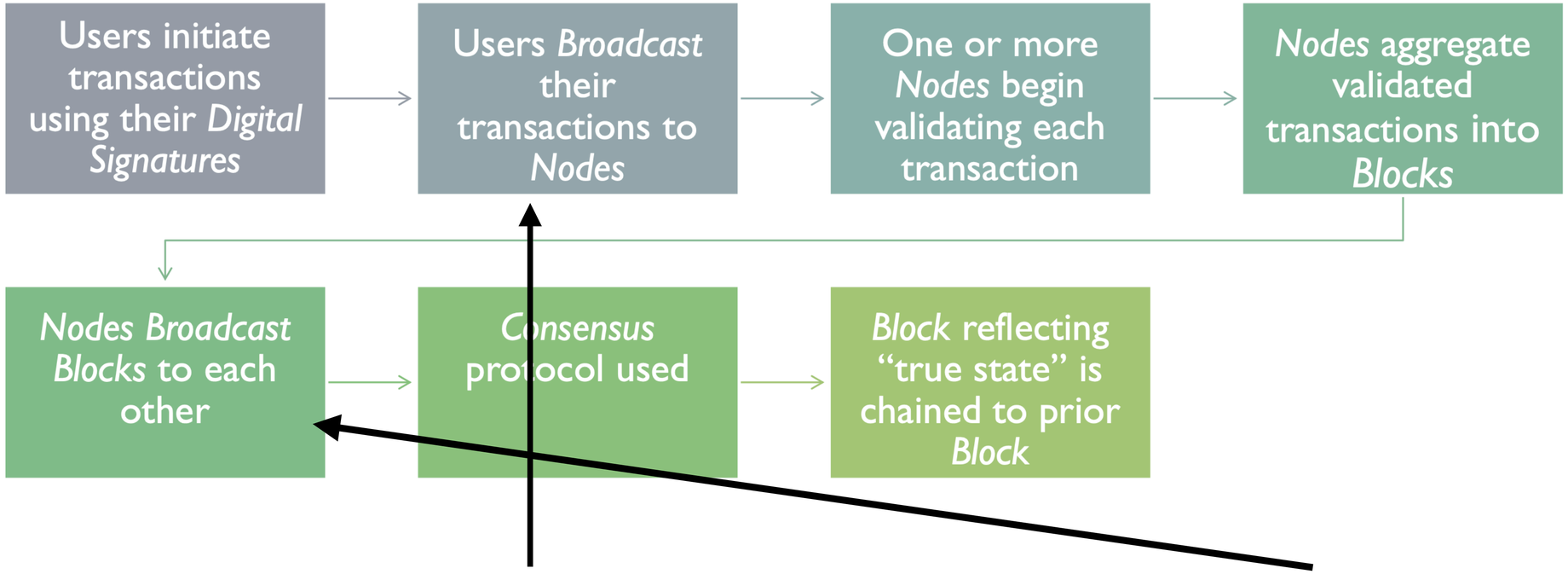
Structured overlay		Unstructured overlay
	Open join and leave?	✪
✪	Minimal overhead?	
	Robust to Attacks, Failures	✪
Maintenance Overhead Hotspots and Load Balancing	Scalable?	Flooding, Spam Redundancy Management

Deep Dive into Layer 0 of Bitcoin Network

Blockchain simplified



Layer 0 questions

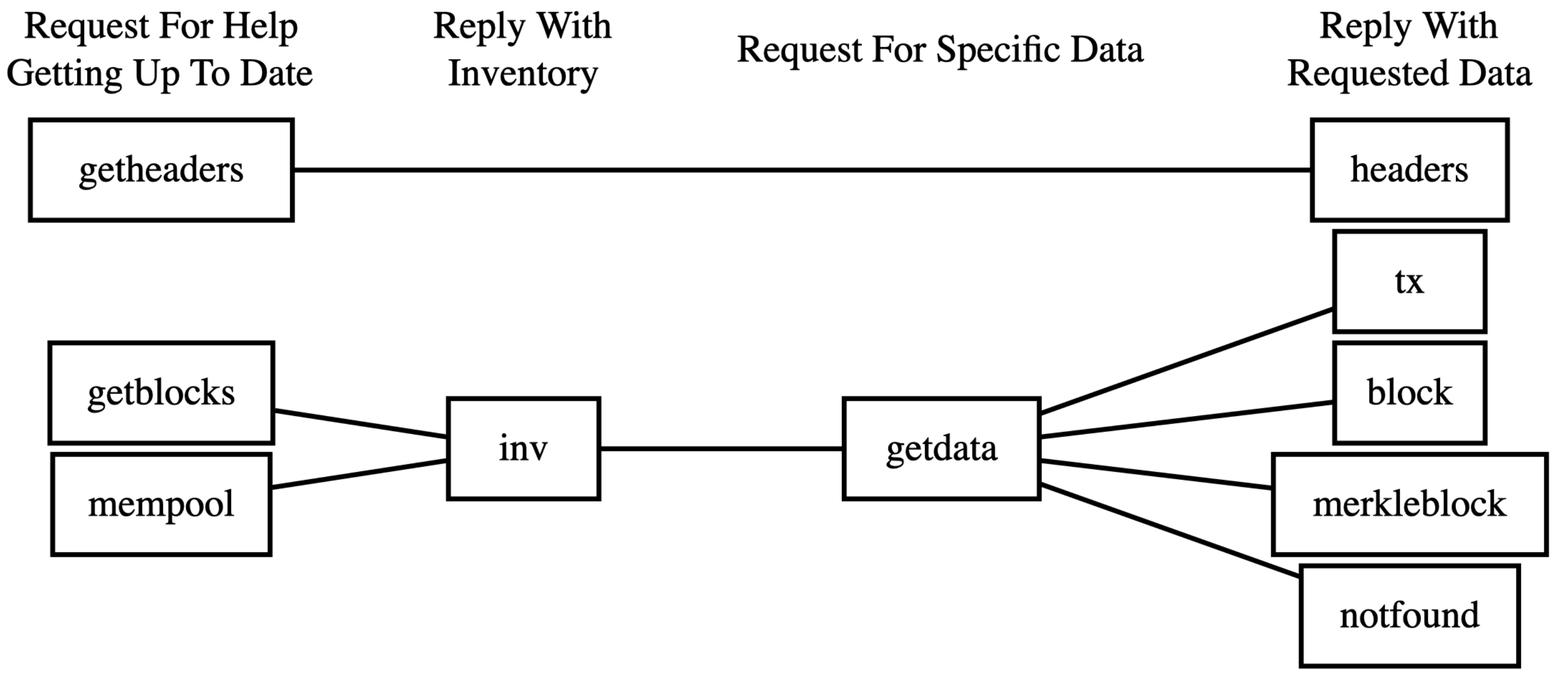


Which nodes?
How many?
All nodes need to see transactions?

How nodes are connected?
Sync all blocks? Or push one?

Bitcoin network messages

Only 4 classes



Overview Of P2P Protocol Data Request And Reply Messages

Same network used for transaction and block propagation

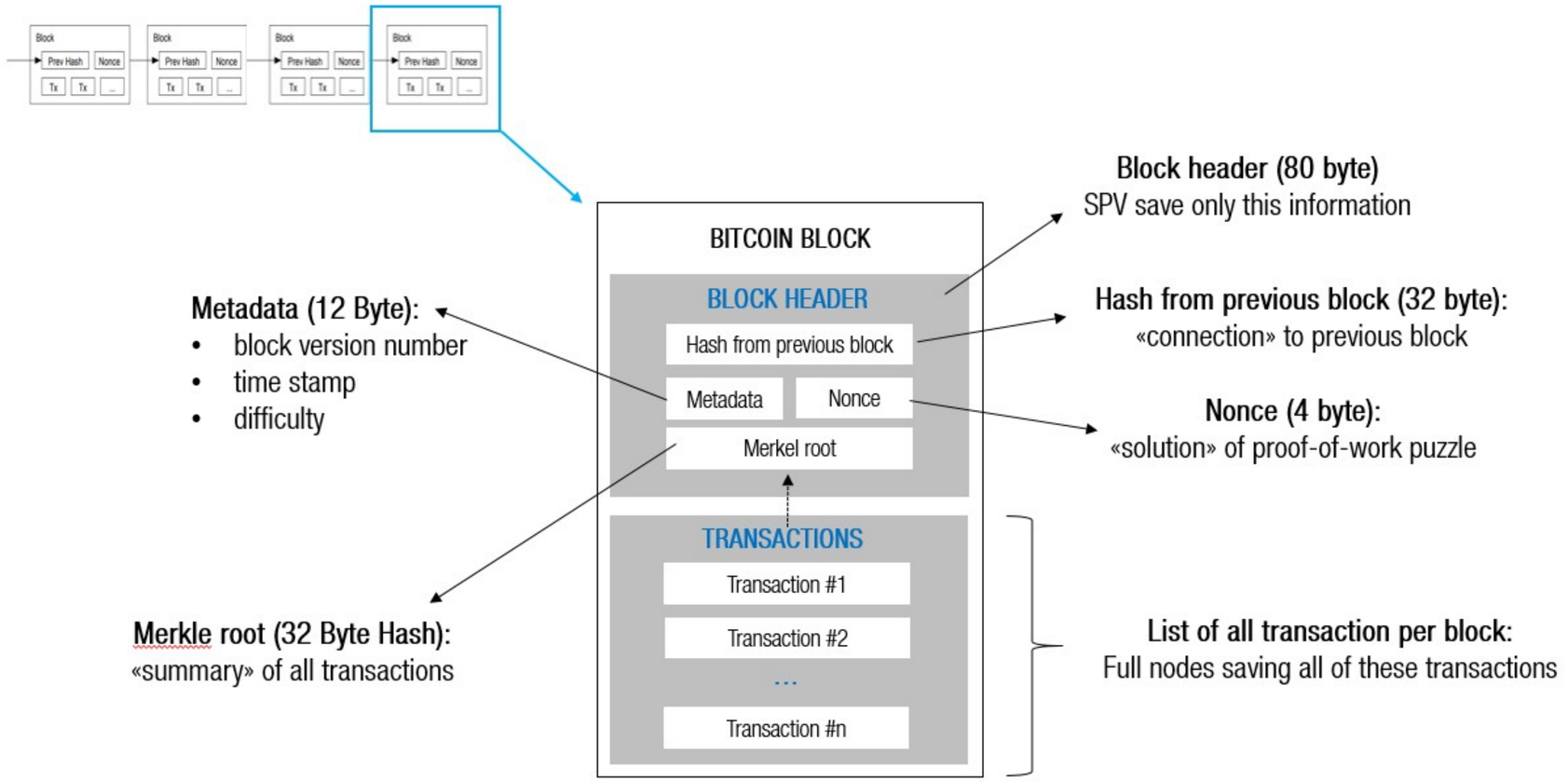
Bitcoin network messages

Messages

- *version* - Information about program version and block count. Exchanged when first connecting.
- *verack* - Sent in response to a version message to acknowledge that we are willing to connect.
- *addr* - List of one or more IP addresses and ports.
- *inv* - "I have these blocks/transactions: ..." Normally sent only when a *new* block or transaction is being relayed. This is only a list, not the actual data.
- *getdata* - Request a single block or transaction by hash.
- *getblocks* - Request an *inv* of all blocks in a range.
- *getheaders* - Request a *headers* message containing all block headers in a range.
- *tx* - Send a transaction. This is sent only in response to a *getdata* request.
- *block* - Send a block. This is sent only in response to a *getdata* request.
- *headers* - Send up to 2,000 block headers. Non-generators can download the headers of blocks instead of entire blocks.
- *getaddr* - Request an *addr* message containing a bunch of known-active peers (for bootstrapping).
- *submitorder*, *checkorder*, and *reply* - Used when performing an IP transaction.
- *alert* - Send a network alert.
- *ping* - Does nothing. Used to check that the connection is still online. A TCP error will occur if the connection has died.

Block Propagation

Two methods of blockchain sync



Block First

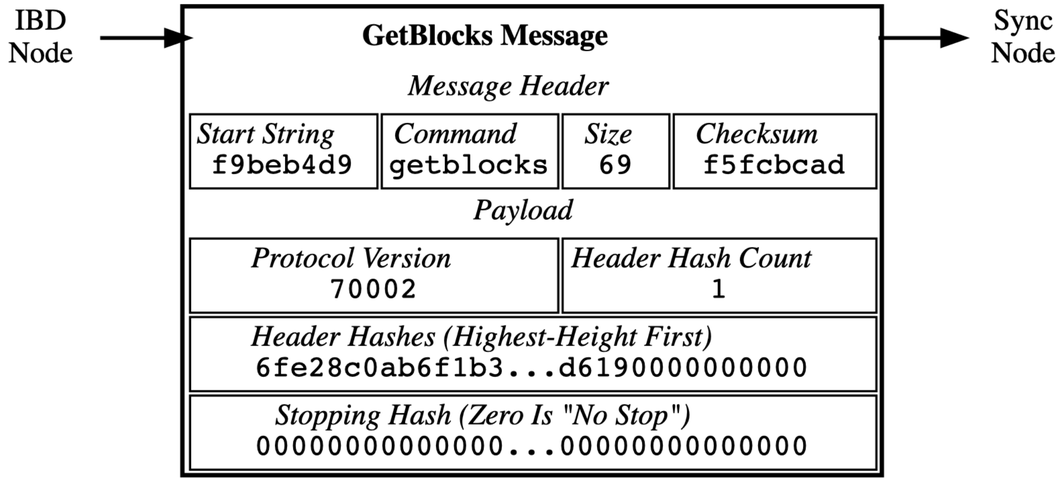
Sequentially download blocks

Headers First

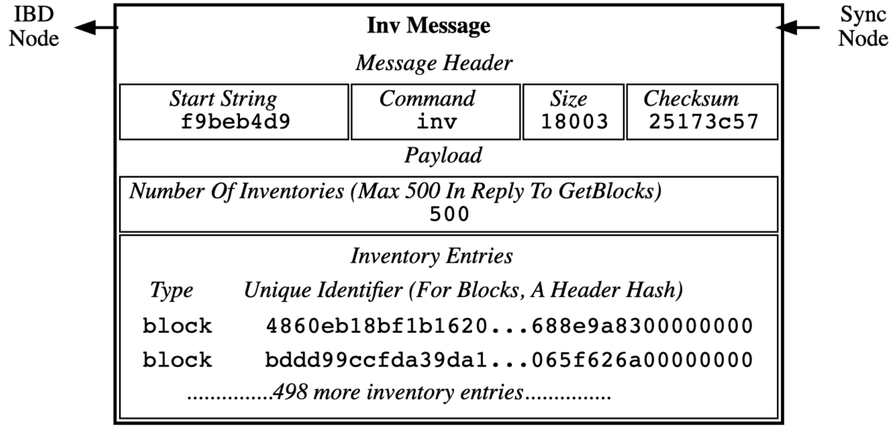
First download all headers

Request block if needed

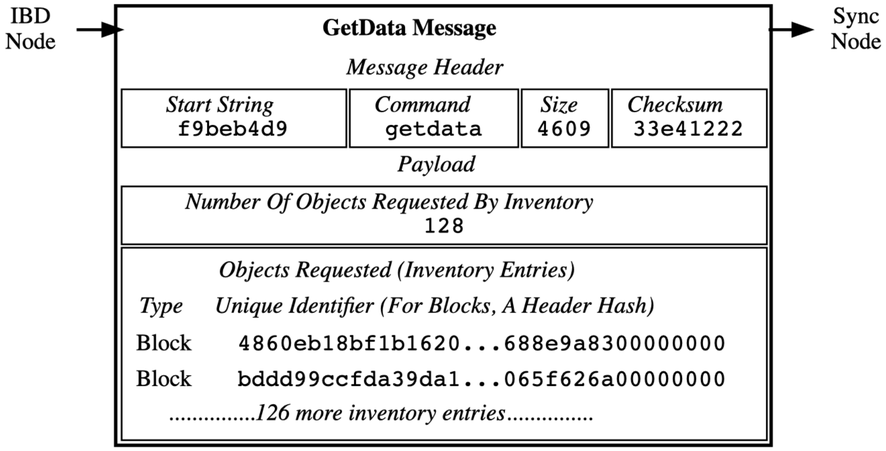
First Block Sync: Introduction Sync



First getblocks message sent from Initial Blocks Download (IBD) node



First inv message reply sent to Initial Blocks Download (IBD) node



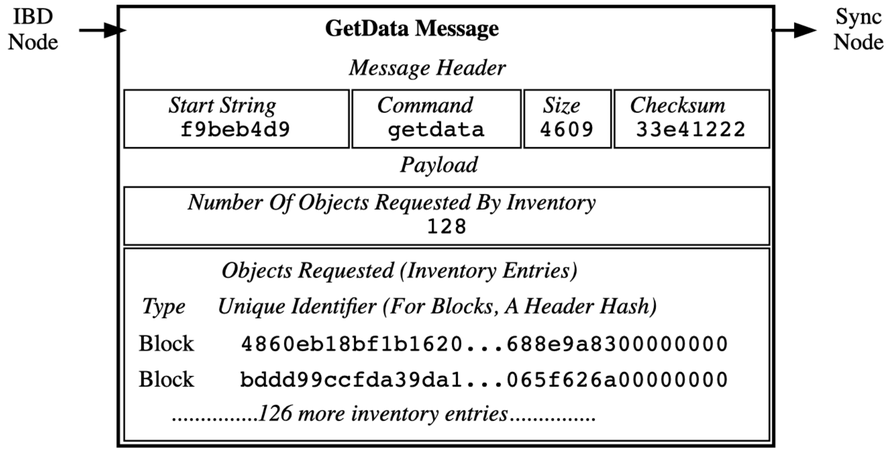
First getdata message sent from Initial Blocks Download (IBD) node

Initial Block Download

Active in Network: Block Broadcasting

Unsolicited Block Push

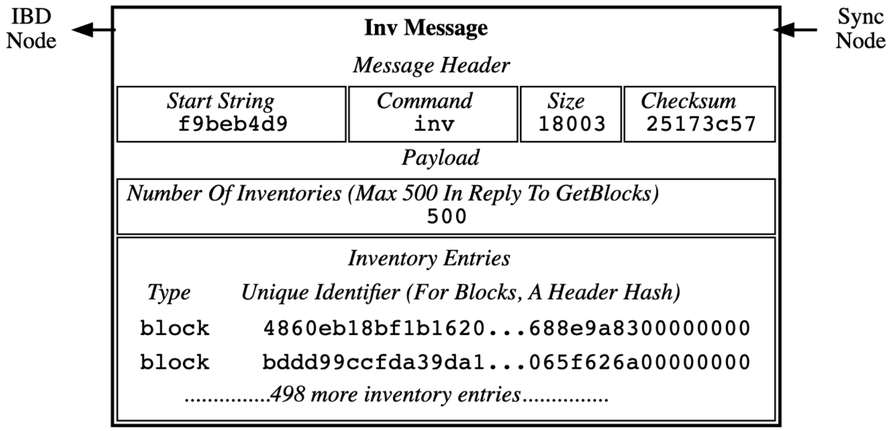
Low Latency
Just block with content
Push Gossip



First getdata message sent from Initial Blocks Download (IBD) node

Standard Relay

High Latency
Additional announcement round
Reconciliation Gossip

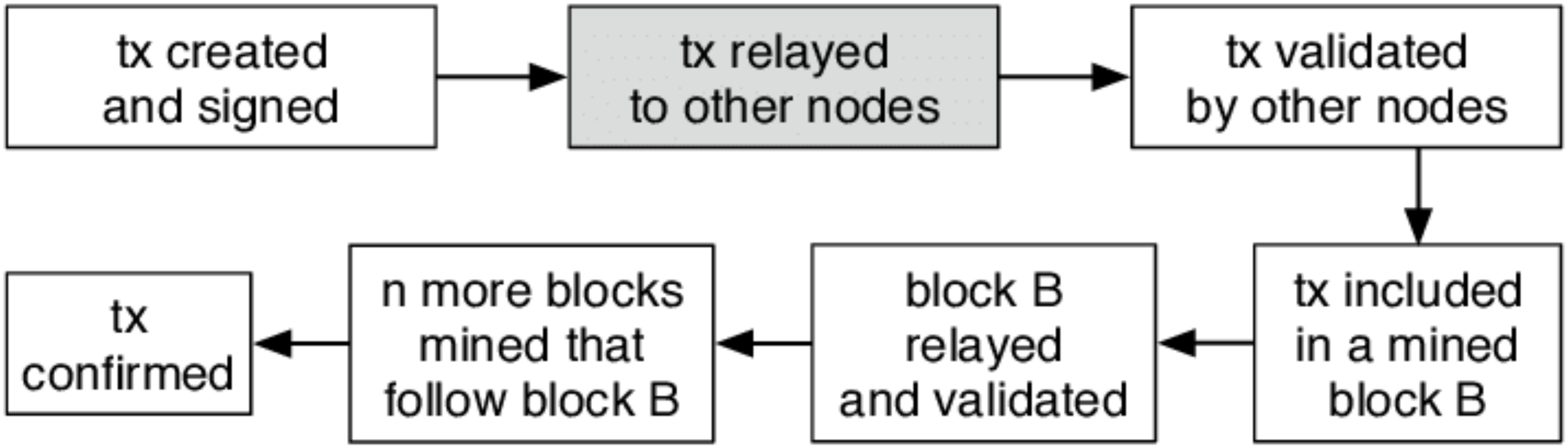


First inv message reply sent to Initial Blocks Download (IBD) node

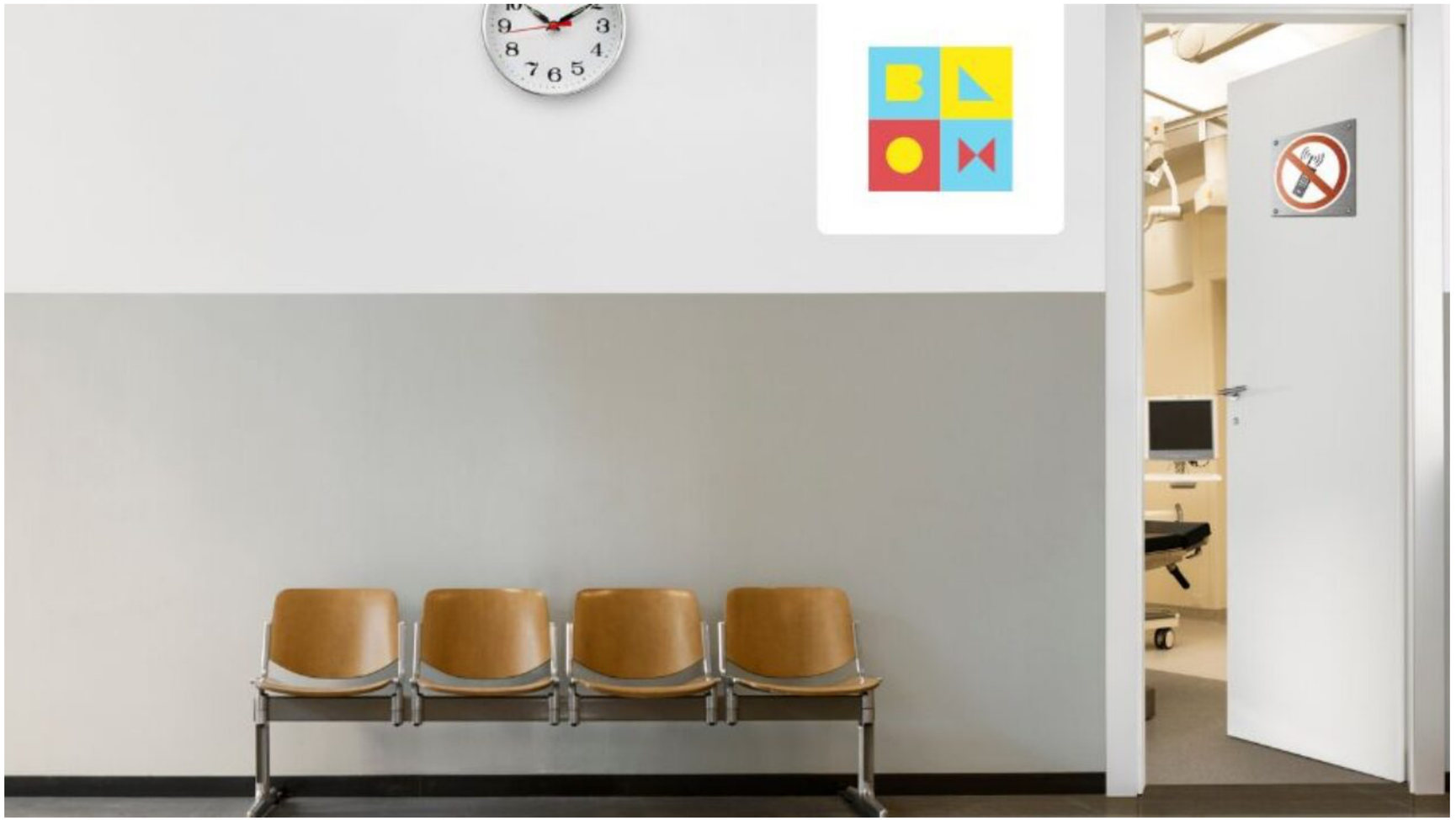
Transaction Propagation

Bitcoin Transaction LifeCycle

Simple story



Transaction Broadcasting: Mempool

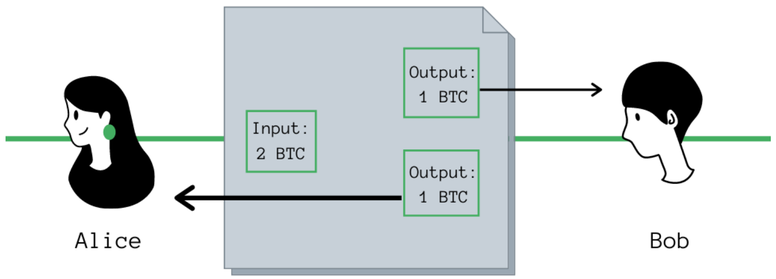


Transaction Broadcasting: Mempool

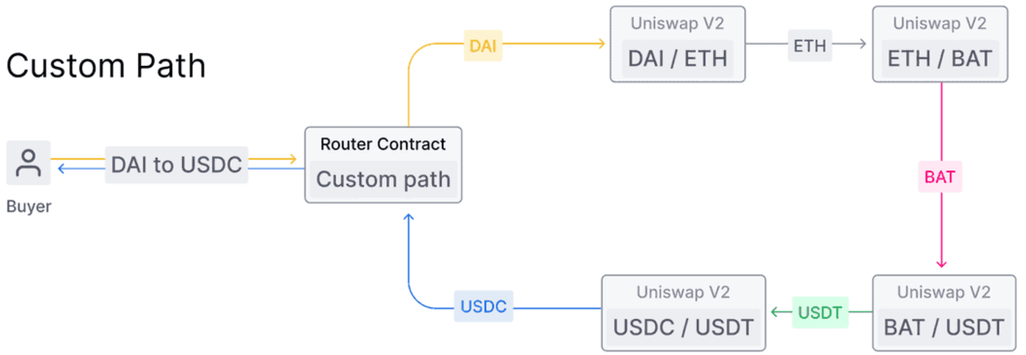
Other Networks

Complexity increase

Ethereum transaction requires more actions

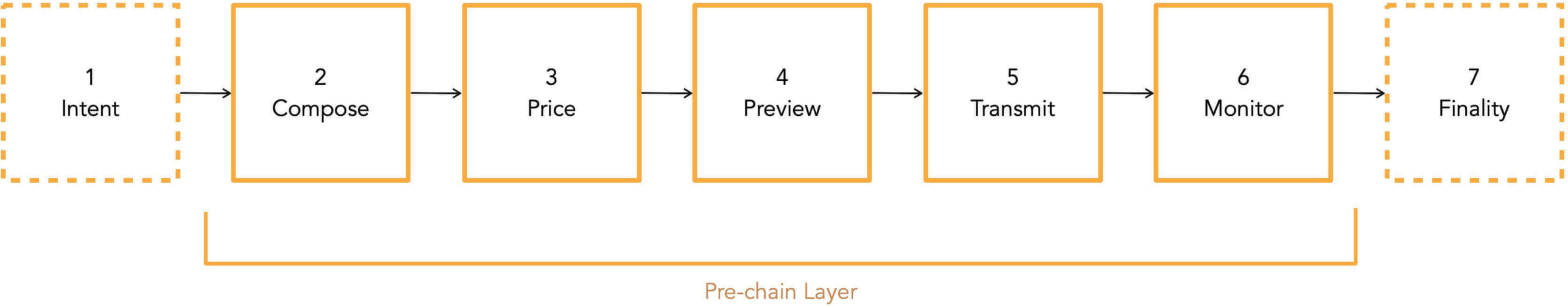


Bitcoin transaction



Ethereum transaction (Uniswap)

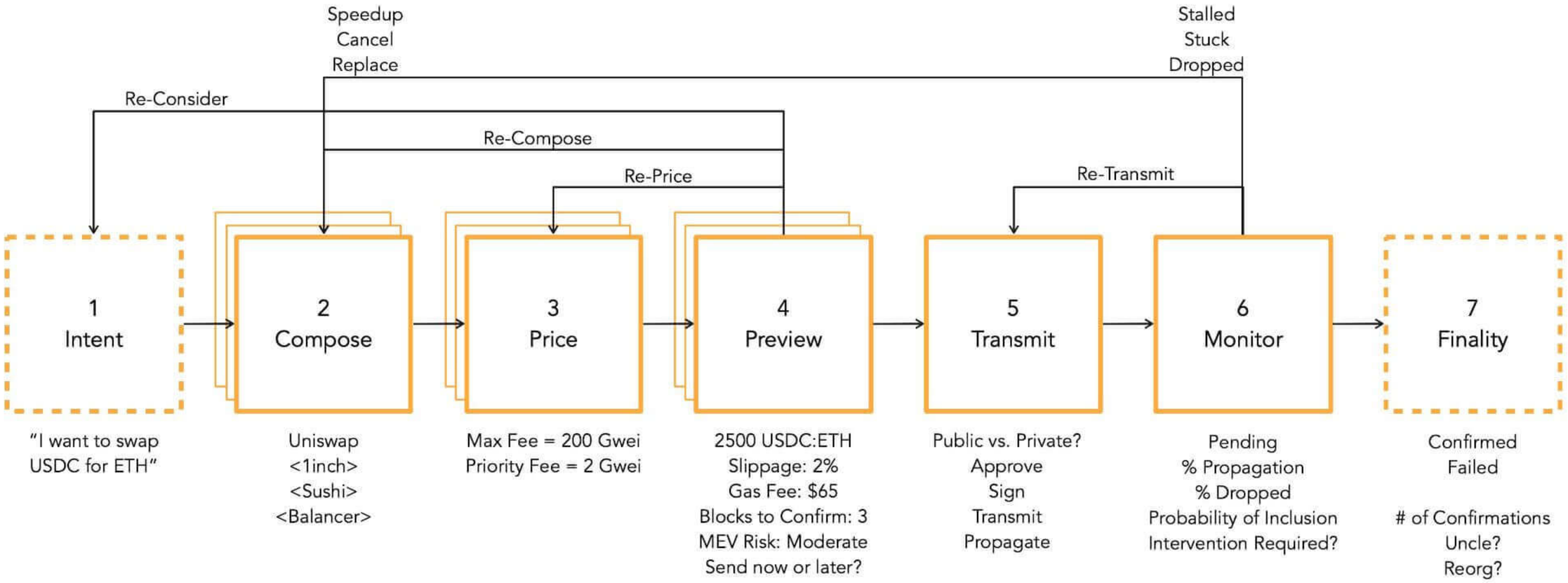
The Web3 Transaction Lifecycle



blocknative

TUDelft

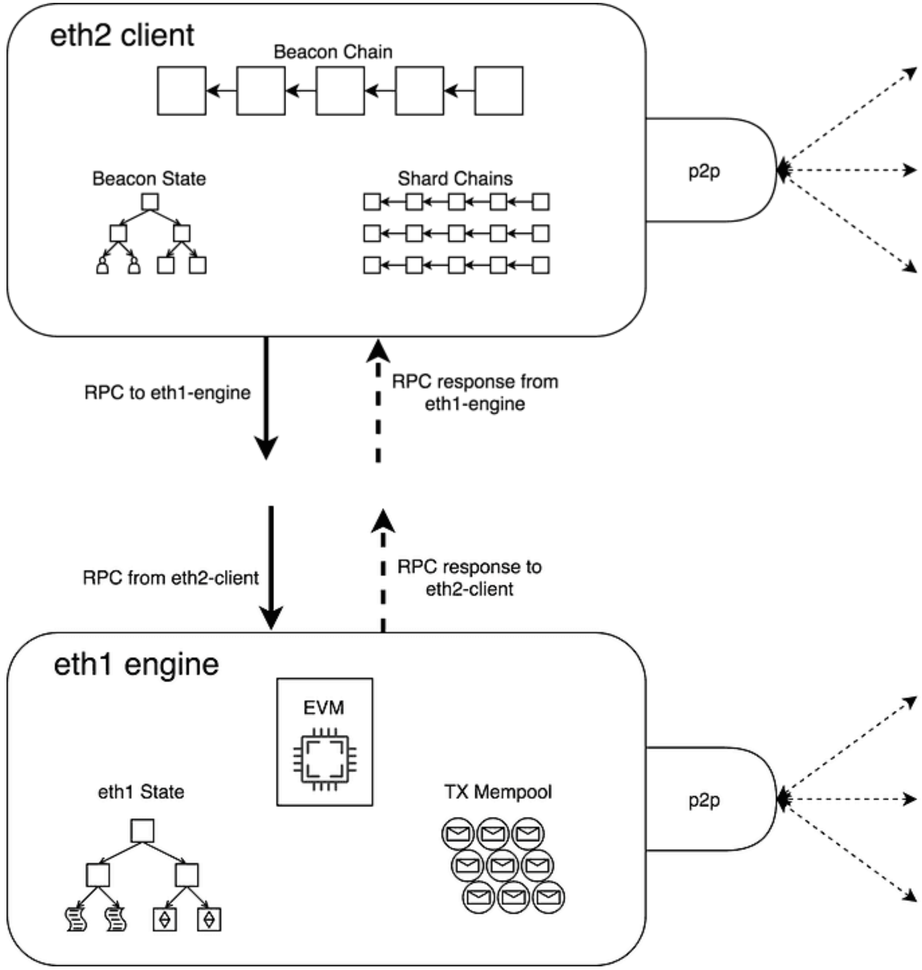
The Web3 Transaction Lifecycle: Not Always Linear



TUDelft

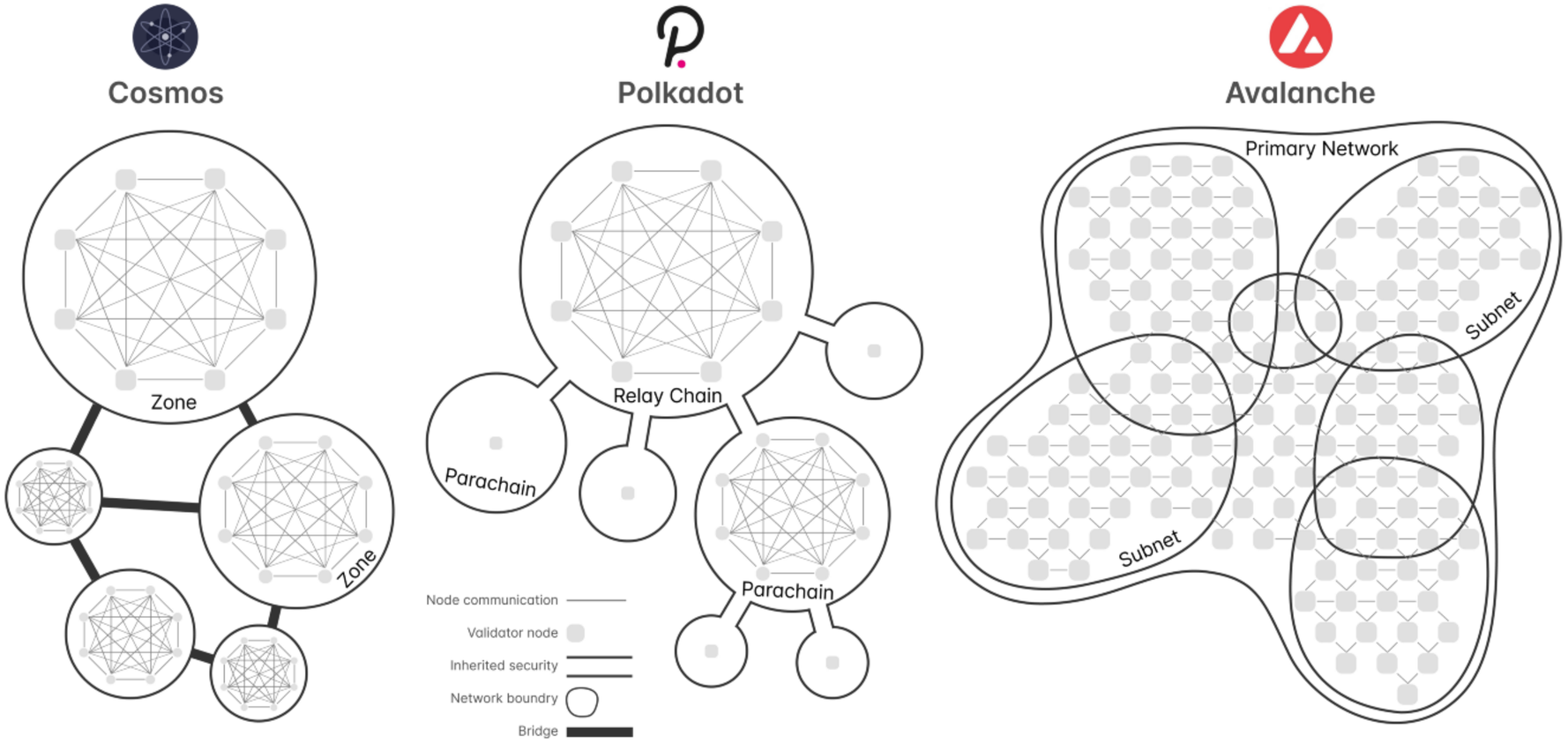
blocknative

Ethereum multiple networks



Multiple subnetworks

More complexity



Demo: Toy Blockchain on Python-IPv8

https://github.com/Tribler/asci-a27/blob/master/src/algorithms/blockchain.py

TU Delft