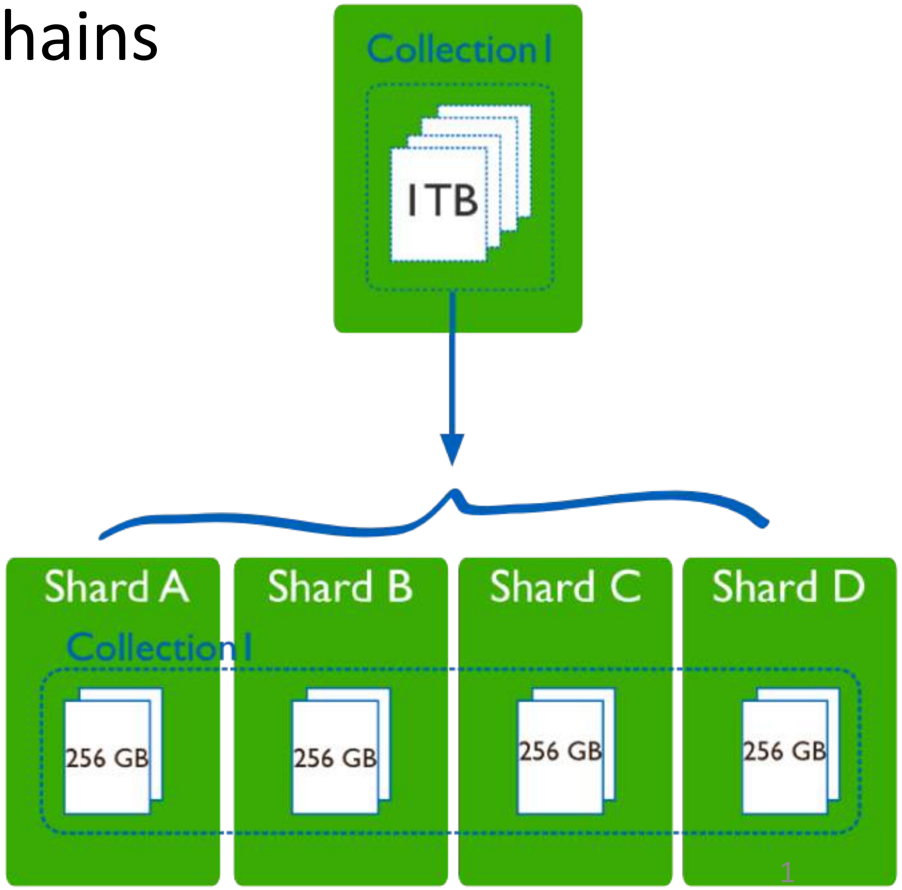
# Advanced Blockchain Storage

Mohammad H. Tabatabaei

UiO : Universitetet i Oslo

# Approaches

- IPFS - Content Addressed, Versioned, P2P File System
- A Secure Sharding Protocol For Open Blockchains



# IPFS (InterPlanetary File System)

- A peer-to-peer distributed file system for connecting all computing devices with the same system of files
- A peer-to-peer hypermedia protocol to make the web faster, safer and more open

# HTTP

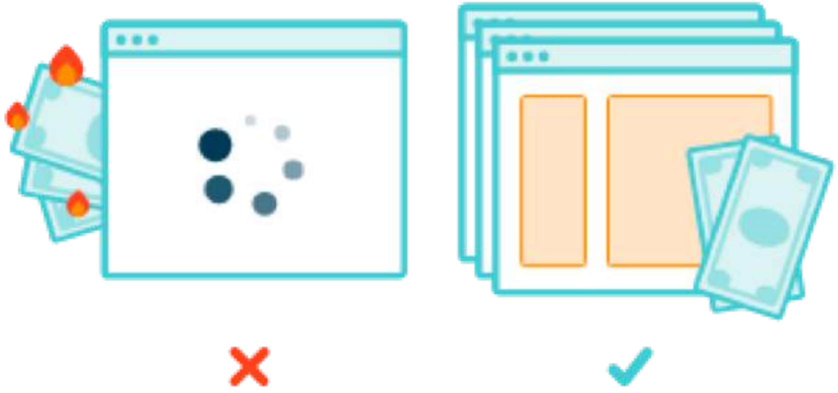
- By far, the most successful distributed system of files ever deployed
- The de facto way to transmit files accross the internet

# New era of data distribution challenges

a. Hosting and distributing petabyte datasets
b. Computing on large data across organizations
c. High-volume high-definition on-demand or real-time media streams
d. Versoning and linking of massive data sets
e. Preventing accidental disappearance of important files

# HTTP is inefficien and expensive

- HTTP downloads a file from a single computer at a time, instead of getting pieces from multiple computers simultaneously.
- IPFS makes it possible to distribute high volumes of data with high efficiency. A P2P approach could save 60% in bandwidth cost.



# Humanity's history is deleted daily

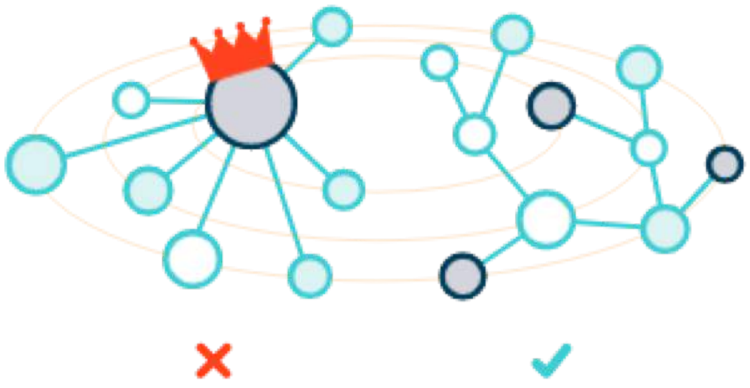
- The average lifespan of a web page is 100 days.

- IPFS provides historic versioning (like git) and makes it simple to set up resilient networks for mirroring of data.

# The web's centralization limits opportunity

- The Internet has been one of the great equalizers in human history and a real accelerator of innovation. But the increasing consolidation of control is a threat to that.

- IPFS remains true to the original vision of the open and flat web, but delivers the technology which makes that vision a reality.



# Internet backbone failure

- IPFS powers the creation of diversely resilient networks which enable persistent availability with or without Internet backbone connectivity.

- IPFS aims to replace HTTP and build a better web.

# What happens when you add file to IPFS (1/2)

1. Each file and all of the **blocks within it** are given a **unique fingerprint** called a **cryptographic hash**.

2. IPFS **removes duplications** across the network and tracks **version history** for every file.

3. Each **network node** stores only content it is interested in, and some indexing information that helps figure out who is storing what.

# What happens when you add file to IPFS (2/2)

4. When **looking up files**, you're asking the network to find nodes storing the content behind a unique hash.

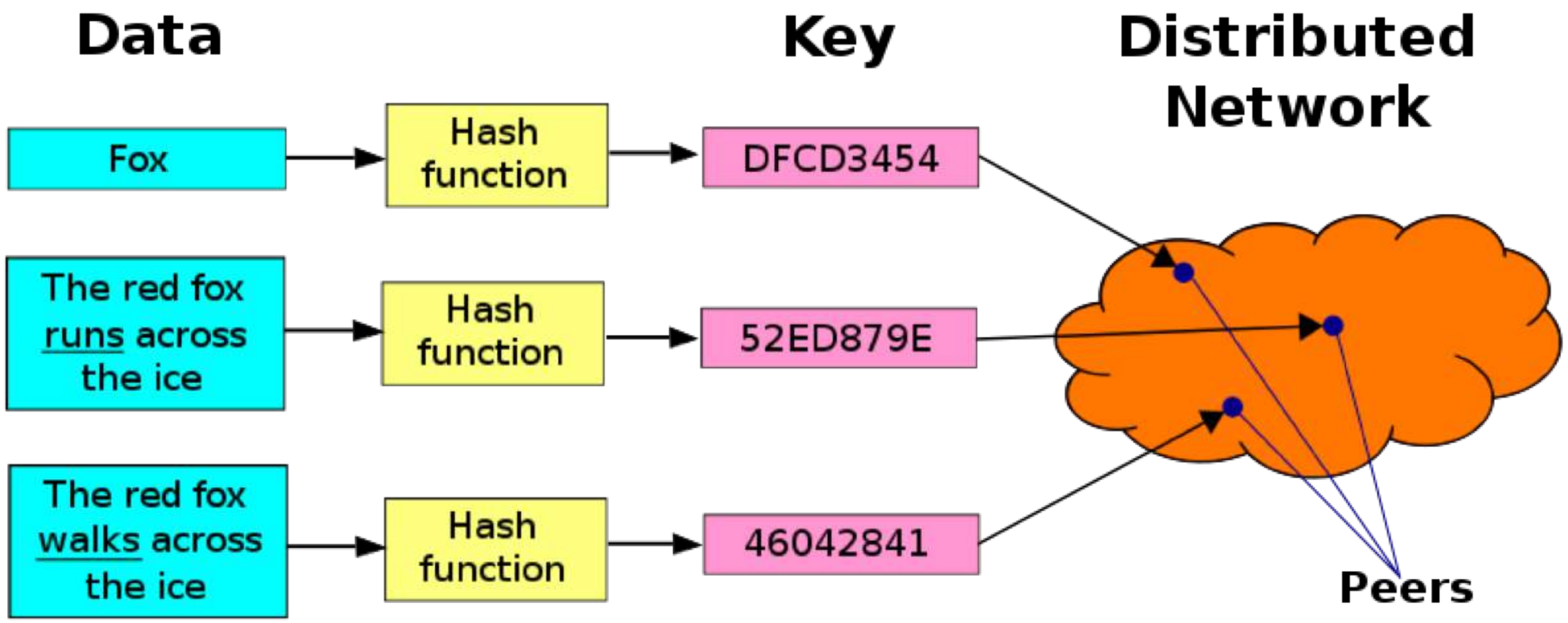

5. Every file can be found by **human-readable names** using a decentralized naming system called **IPNS**.

# IPFS Underlying Technologies

1. Distributed Hash Tables
2. Block Exchange – BitTorrent
3. Version Control Systems – Git
4. Self-Certified Filesystems - SFS

# DHT (Distributed Hash Tables)

- If you have the key, you can retrieve the value
- But the data is distributed over multiple nodes



# DHT implementations

- **Kademlia**
  - The DHT protocol that is used in almost all popular P2P systems
  - Uses the ID of the nodes to get step by step closer to the node with the desired hash
- **Coral DSHT**
  - Improves the lookup performance and decreases resource use
- **S/Kademlia DHT**
  - Makes Kademlia more resistant against malicious attacks

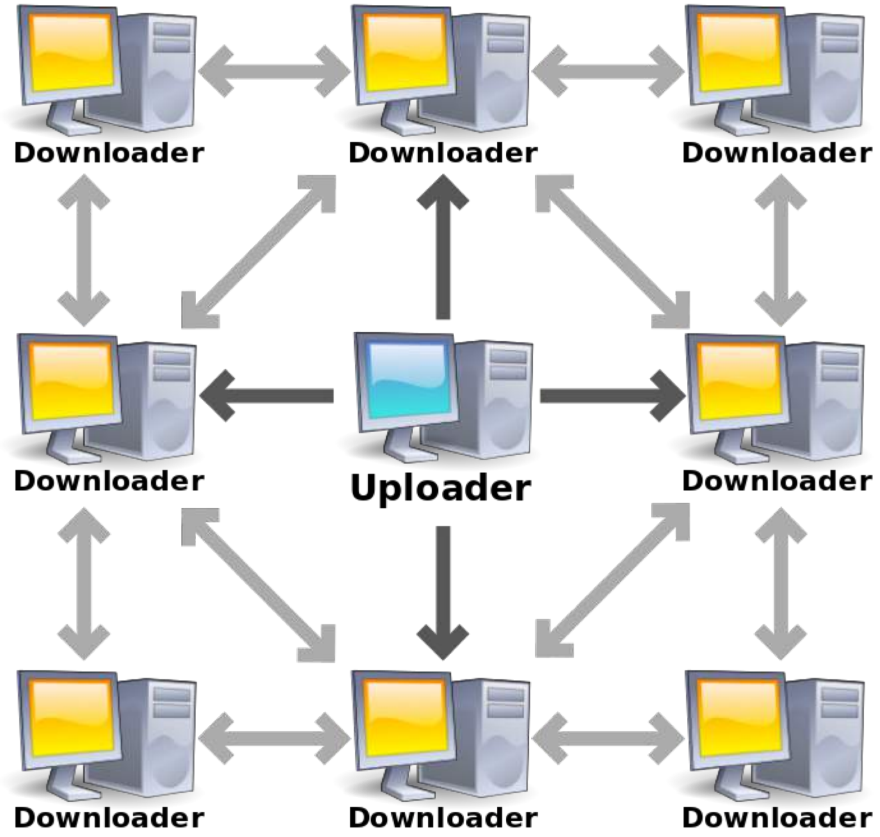
# DHT in IPFS

- In the case of IPFS, the key is a hash over the content.
- Ask an IPFS node for the content with hash QmcPx9ZQboyHw8T7Afe4DbWFcJYocef5Pe4H3u7eK1osnQ
- The IPFS node will lookup in the DHT which nodes have the content.

- The DHT is used in IPFS for **routing**:
  1. to announce added data to the network
  2. and help locate data that is requested by any node.

# Block Exchanges - BitTorrent

- A P2P filesharing stsyem which helps networks of untrusting peers (swarms) to cooperate in dirstributing pieces of files to each other.



# BitTorrent in IPFS

- Two BitTorrent features that IPFS uses:
  1. Tit-for-tat strategy (if you don't share, you won't receive either)
  2. Get rare pieces first

- Difference:
  - In BitTorrent each file has a separate swarm of peers where IPFS is one big swarm of peers for all data.

- The IPFS BitTorrent variety is called **BitSwap**.

# Version Control Systems - Git

- Provide facilities to model files changing over time and distribute different versions efficiently.
- Git, the popular version control system:
  - Git only adds data, so objects are immutable.
  - Git hashes the content with SHA1 and uses these hashes in its database not the file or directory name.
  - Links to other objects are embedded, forming a Merkle DAG which provides many useful integrity and workflow properties.

# Self-Certified Filesystems - SFS

- Allows generating an address for a remote filesystem, where the user can verify the validity of the address.

- Using the following scheme: /sfs/<Location>:<HostID>

- where Location is the server network address, thus the name of an SFS file system certifies its server.

# IPFS Design (Sub-protocols)

1. **Identities:** name the nodes
2. **Network:** Talke to other clients
3. **Routing:** Announce and find stuff
4. **Exchange:** Give and take
5. **Objects:** Organize the data
6. **Files:** Versioned file system hiererchy
7. **Naming:** A self-certifying mutable name system

# Identities (1/2)

- Users are free to instantiate a new node identity on every launch:
  1. generate a PKI key pair (public + private key)
  2. hash the public key
  3. the resulting hash is the NodeId

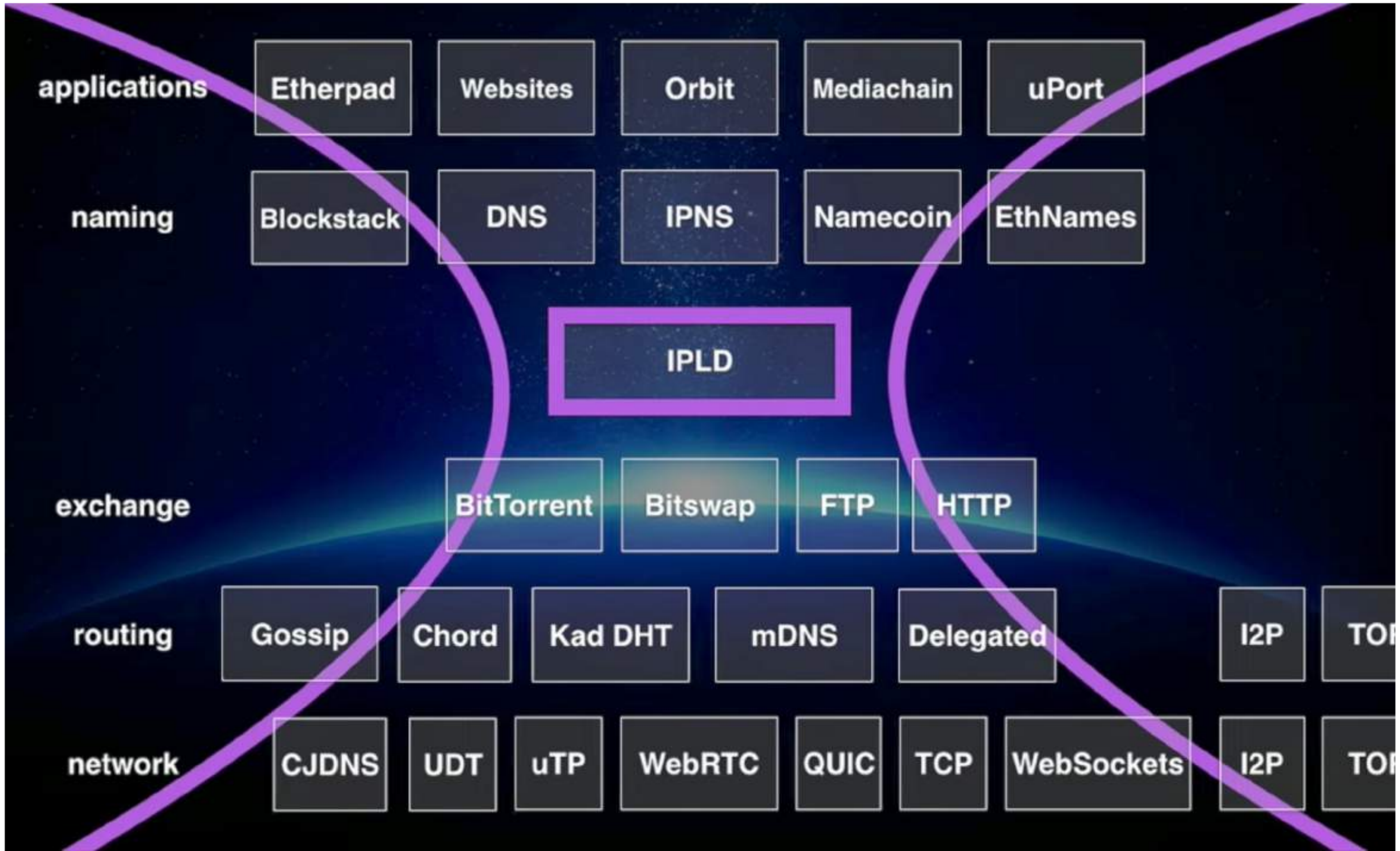
```
type Node struct {
  NodeId NodeID
  PubKey PublicKey
  PriKey PrivateKey
}
```

# Identities (2/2)

- When two nodes start communicating the following happens:
  1. exchange public keys
  2. check if: hash(other.PublicKey) == other.NodeId
  3. if so, we have identified the other node and can e.g. request for data objects
  4. if not, we disconnect from the "fake" node

# Network

- IPFS works on top of any network.



# Routing

- The routing layer is based on a DHT and its purpose is to:
  - announce that this node has some data, or
  - find which nodes have some specific data, and
  - if the data is small enough (=< 1KB) the DHT stores the data as its value.

# Exchange (1/2)

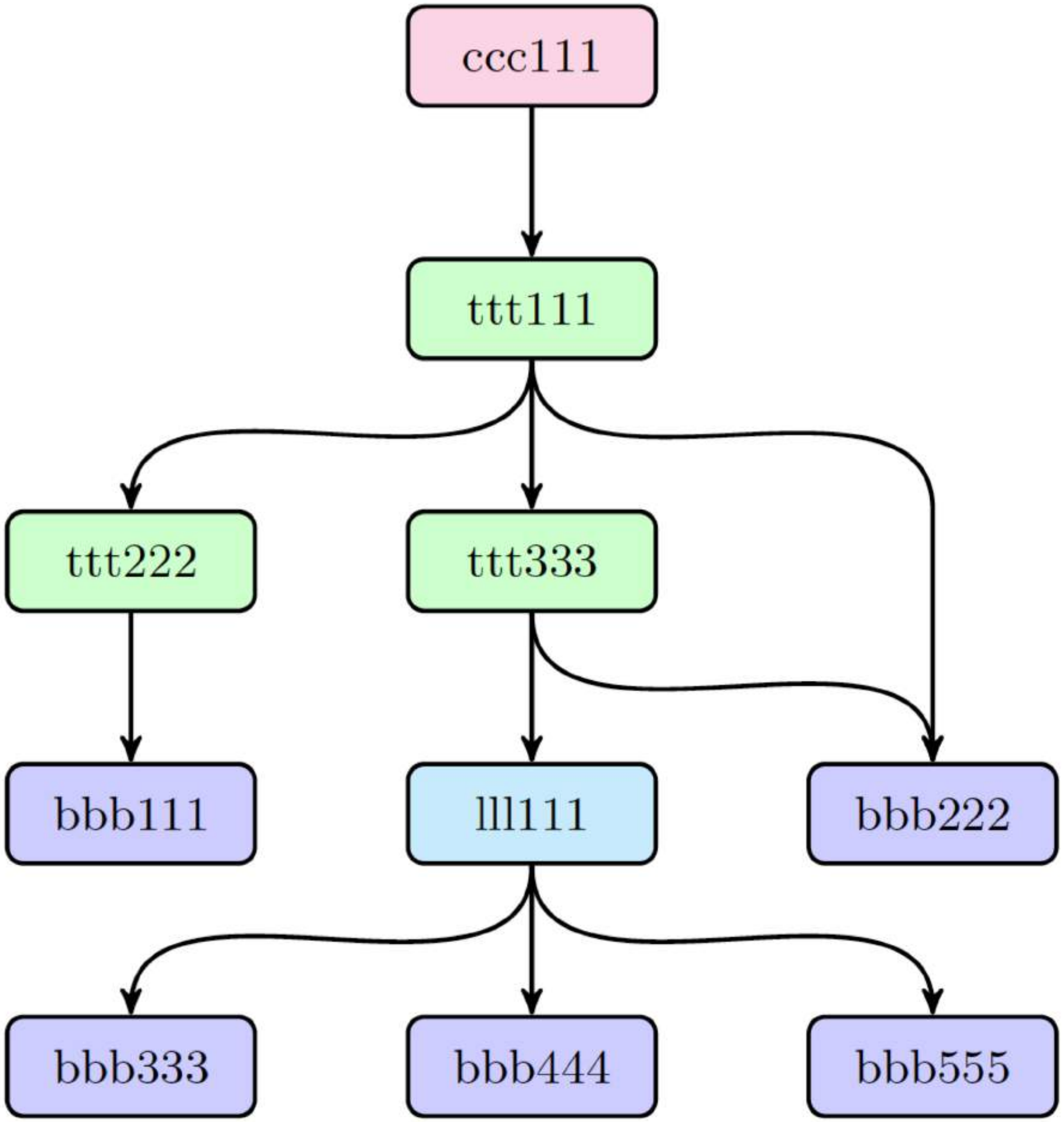
- Data is broken up into blocks, and the exchange layer is responsible for distributing these blocks.
- When peers connect, they exchange which blocks they have (have_list) and which blocks they are looking for (want_list)
- To decide if a node will actually share data, it will apply its BitSwap Strategy
  - Tit-for-tat
  - BitTyrant: Sharing the least possible
  - BitThief: Never share
  - PropShare: Sharing proportionally

# Exchange (2/2)

- When peers exchange blocks they keep track of the amount of data they share (credit) and the amount of data they receive (debt).
- They keep track of history in the BitSwap Ledger.
- If a peer has credit (shared more than received), our node will send the requested block.
- If a peer has debt, our node will share or not share.

# Objects & Files

- IPFS uses hash-linked data structure:
- Organize the data in a graph, where we call the nodes of the graph ***objects***.
- These objects can contain data and/or links to other objects.
- These links - Merkle Links - are simply the cryptographic hash of the target object.



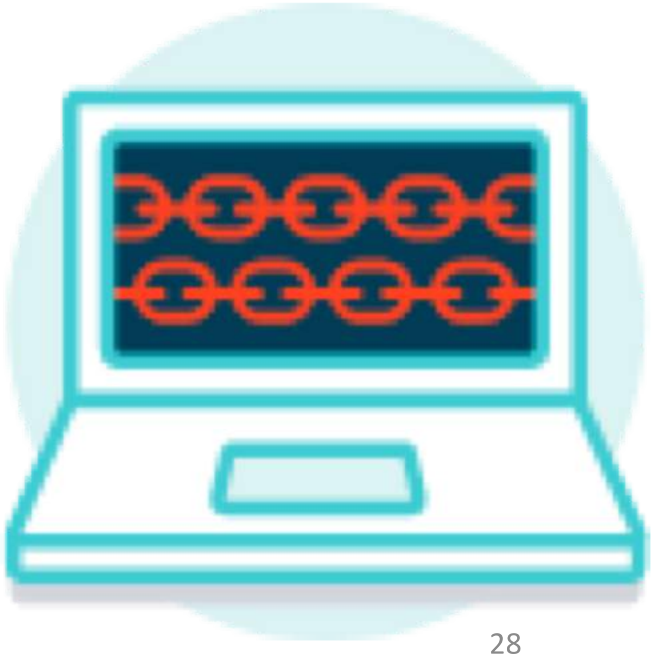
# Object Merkle DAG Advantages

- Objects can be:
  a. Retrieved via their hash
  b. Integrity checked
  c. Linked to others
  d. Cached indefenitiley

- Objects are permanent.

# IPFS & Blockchain

- You can address large amounts of data with IPFS, and place the immutable, permanent IPFS links into a blockchain transaction.
- This timestamps and secures the content, without having to put the data on the chain itself.



# A Secure Sharding Protocol For Open Blockchains

Loi Luu
National University of Singapore
loiluu@comp.nus.edu.sg

Viswesh Narayanan
National University of Singapore
visweshn@comp.nus.edu.sg

Chaodong Zheng
National University of Singapore
chaodong.zheng@comp.nus.edu.sg

Kunal Baweja
National University of Singapore
bawejaku@comp.nus.edu.sg

Seth Gilbert
National University of Singapore
seth.gilbert@comp.nus.edu.sg

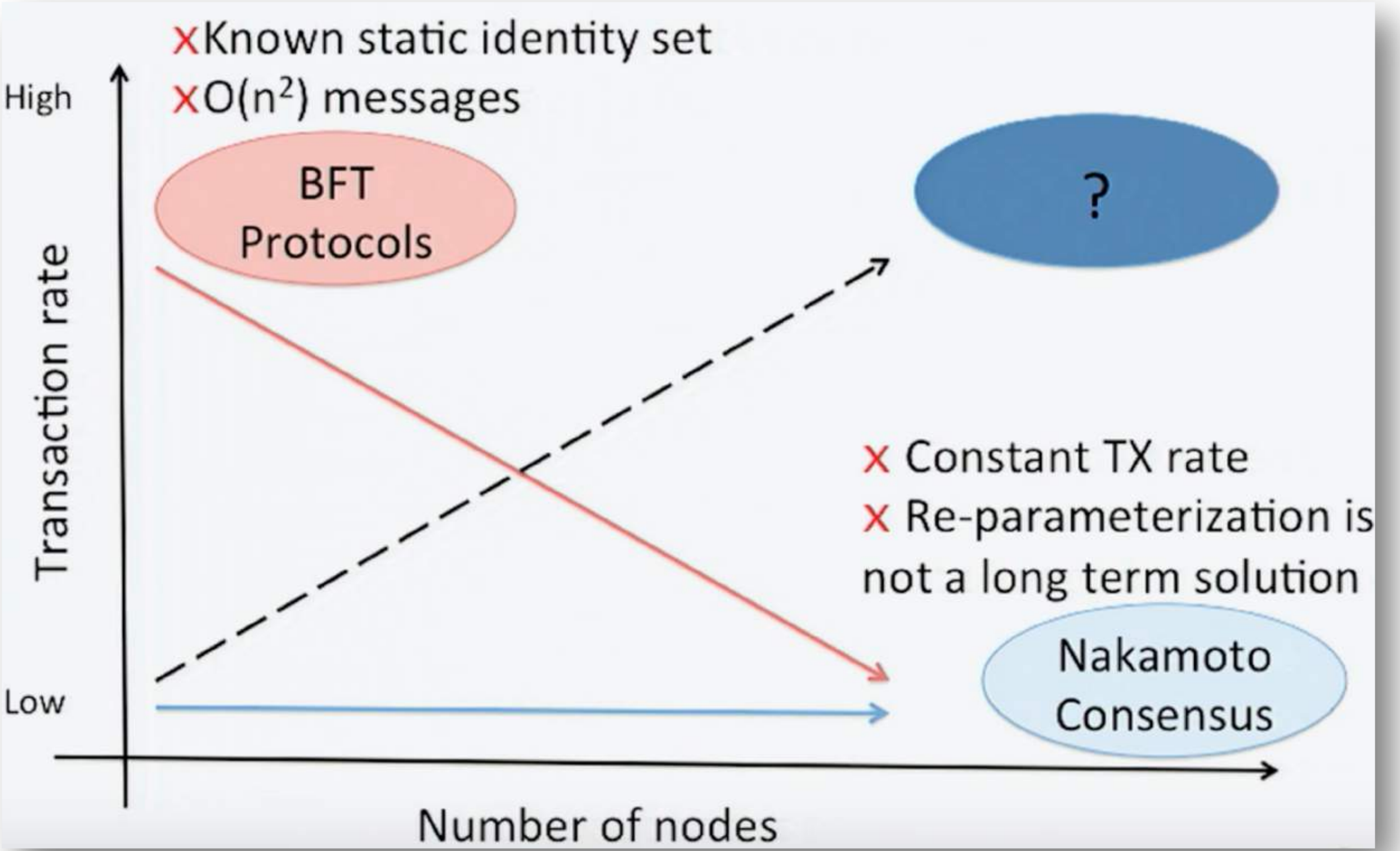
Prateek Saxena
National University of Singapore
prateeks@comp.nus.edu.sg

# Scalability Issue

- Bitcoin:
  - 1 MB block per 10 mins
  - 3-7 transactions per second

- Demand from practice: 1200-50000 transactions per second

# Existing protocols are not scalable



# Elastico Contributions

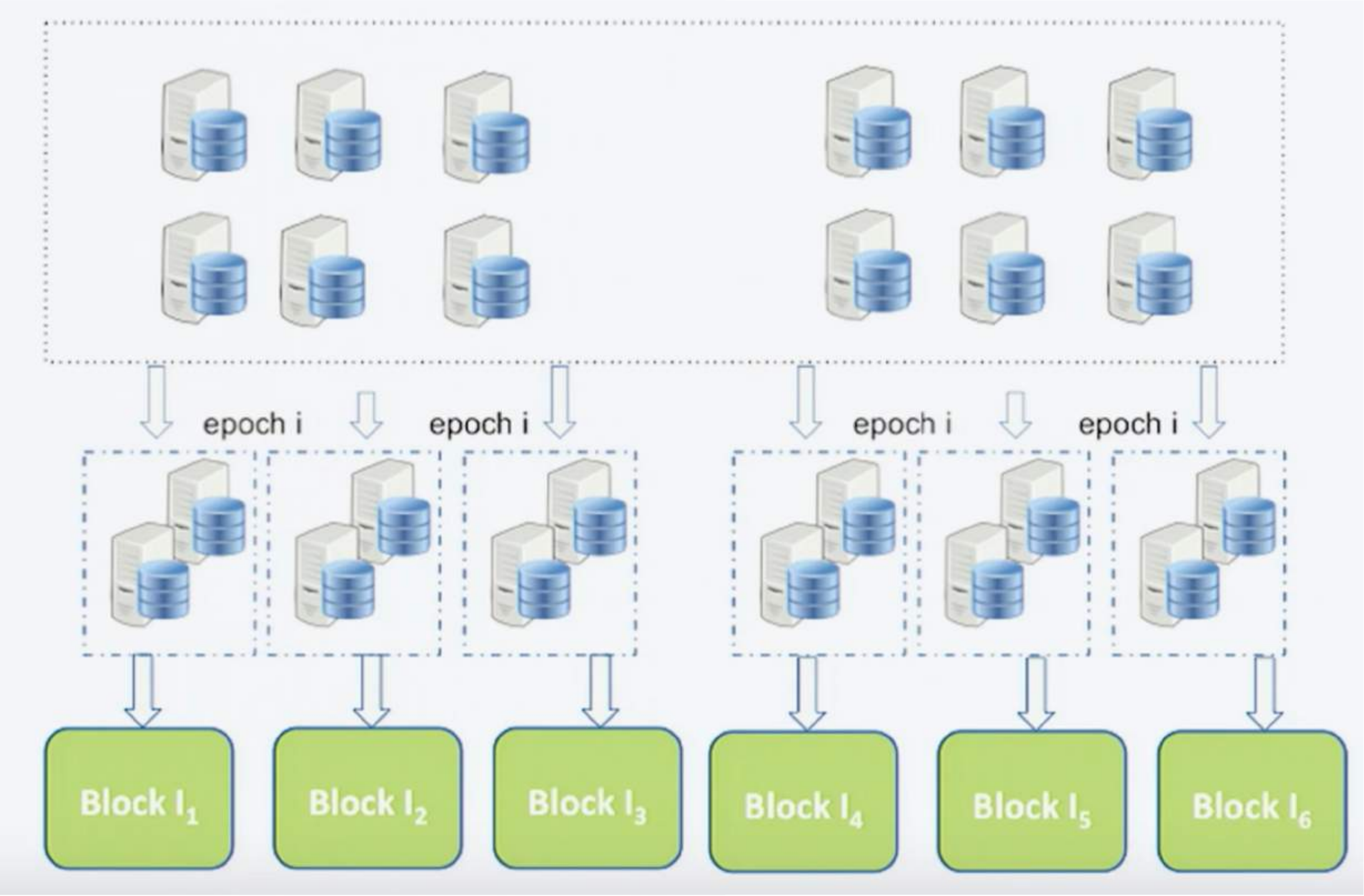
- Near-linear computational scalability
- Tolerate up to 25% adversary
- Secure sharding in open networks

# Problem Statement & Assumptions

- Problem:
  - Agree on O(N) blocks per epoch
  - Cost per node stay constant

- Assumptions:
  - Synchronous network
    - Bounded delay from a node to all other honest nodes
  - At most 1/4 computation power is controlled by adversary
  - Nodes have equal computation power

# Concept of Sharding: More Nodes, More Transactions Blocks

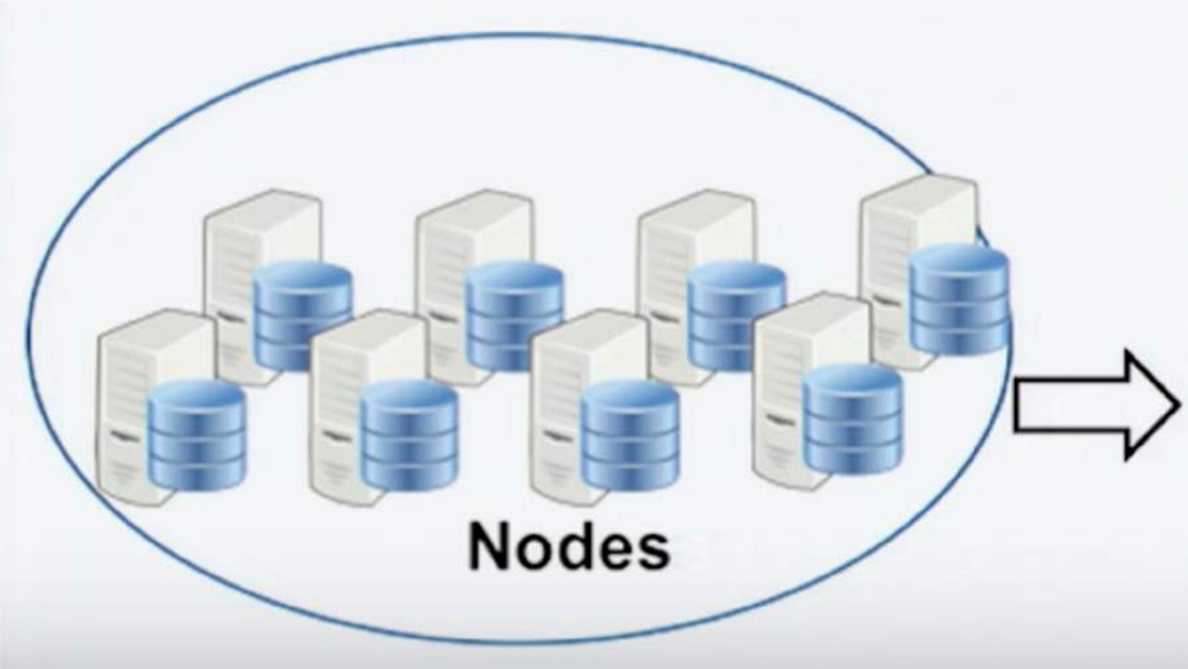


# Elastico Solution

1. Identity Establishment and Committe Formation
    - Use PoW to estabilsh identity
2. Overlay Setup for Committees
    - Communicate to discover others in their committee
3. Intra-committee Consensus
    - Run PBFT within their committee to agree on a single set of transactions
4. Final Consensus Broadcast
    - Compute all the values received frim all the committees
5. Epoch Randomness Generation
    - Generate a set of different random numbers for the PoW of the next epoch

# Step 1: Identity Establishment

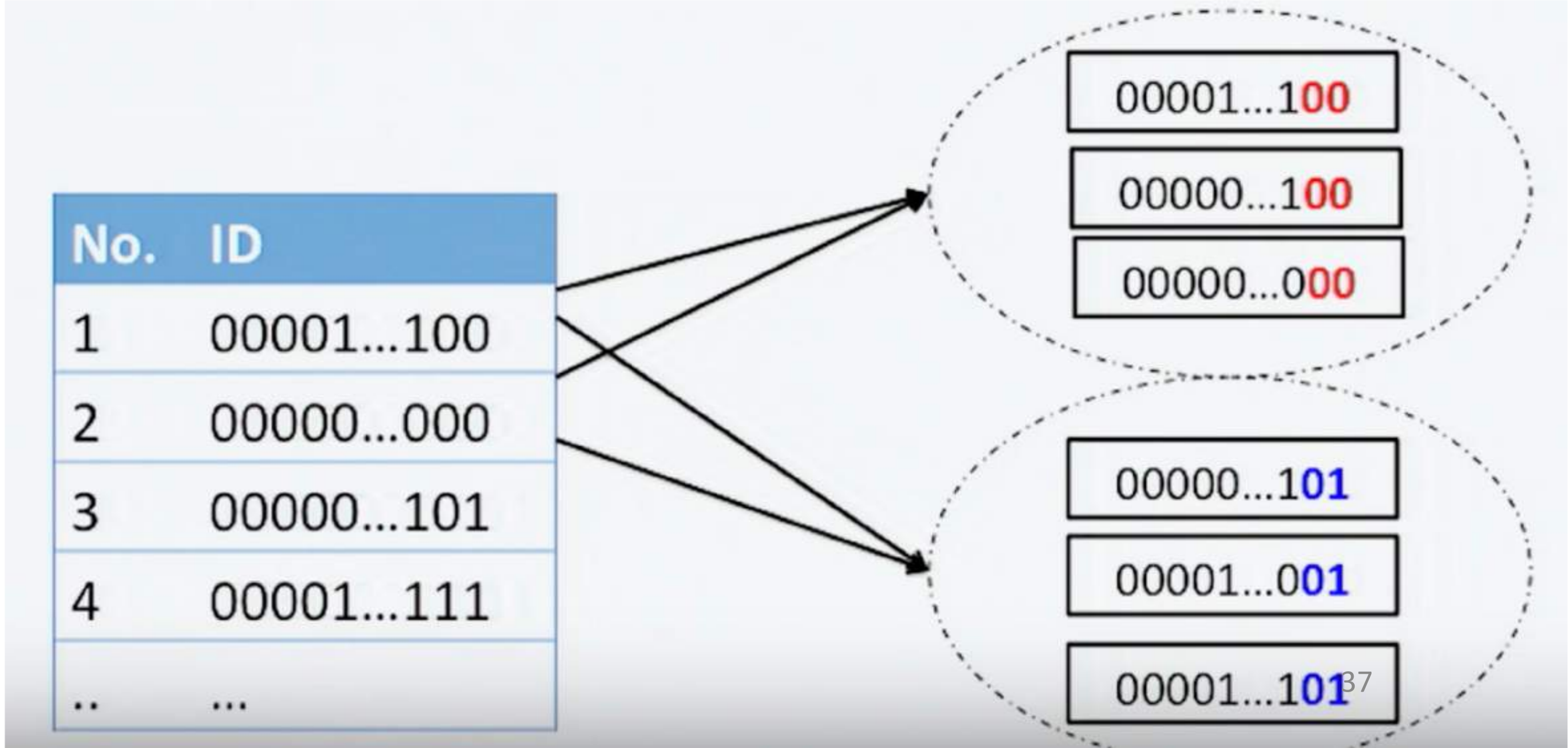
- Solve PoW
  - ID = H ( EpochRandomness, IP, Pubkey, Nonce) < D




| No. | ID | IP | Pubkey |
|---|---|---|---|
| 1 | 000011...01 | 1.2.3.4 | 0x01aaff.. |
| 2 | 000010...10 | 5.6.7.8 | 0x02adff.. |
| .. | ... | ... | .... |

# Step 2: Assigning Committees

- Goals:
  - Fairly distribute identities to committees
  - Guarantee at most 1/3 malicious
- Use last K bits of the ID

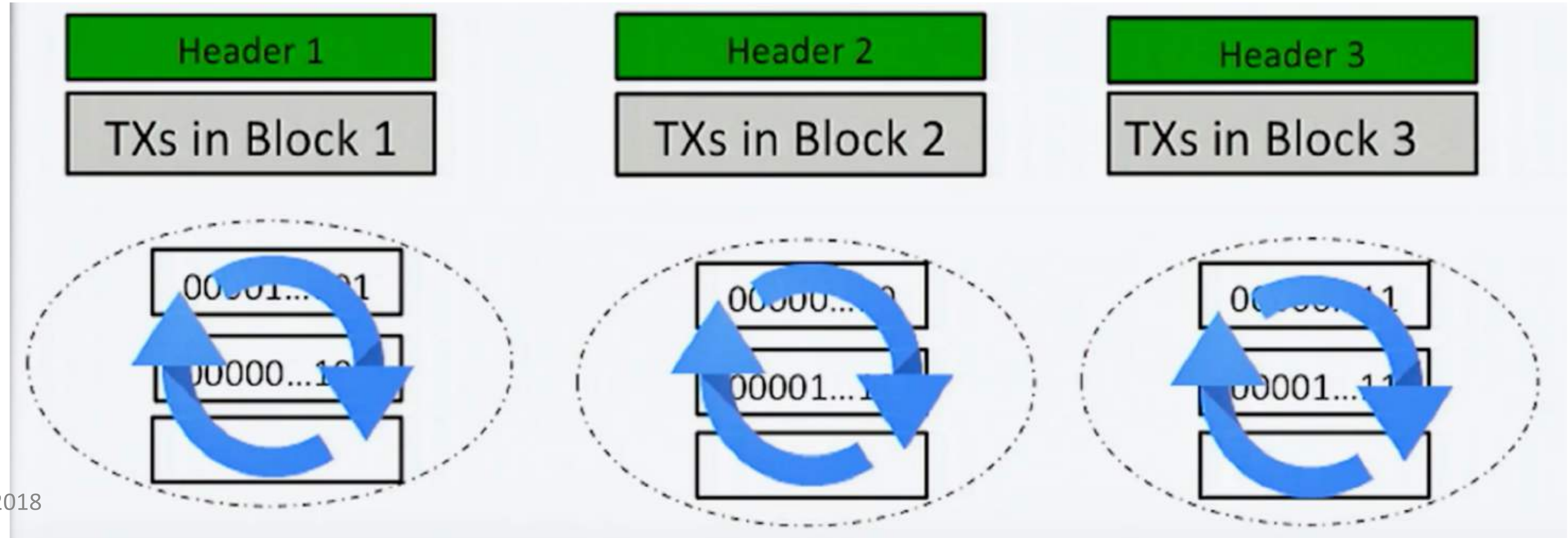


# Use Directory Committee

1. First C identities become directory servers
2. Latter nodes sends IDs to directories
3. Directories send committee list to nodes
4. No. messages: O(NC)

# Step 3: Propose a Block within a committee

- Run a classical byzantine agreement protocol
  - Members agree and sign on one valid data block
  - Number of messages = O ($C^2$)
  - Valid data blocks have 2C/3+1 signatures



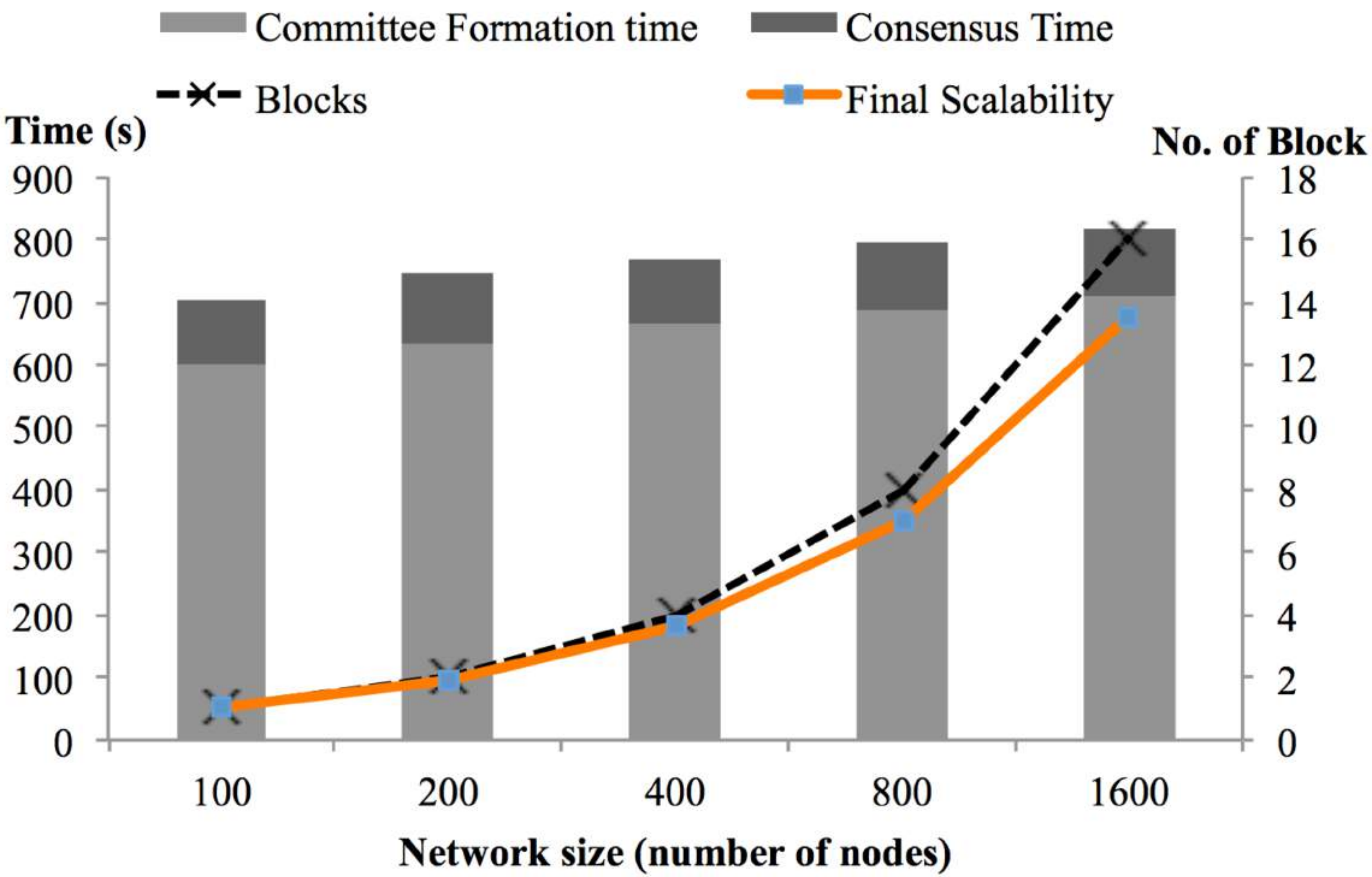
# Step 4: Final Committee unions all blocks

- Each committee send their block header to the Final Committee
- Final Committee runs BFT protocol to produce final block
- Then broadcast final block to everyone

# Step 5: Generate Epoch Randomness

- ID = H ( EpochRandomness, IP, Pubkey, Nonce) < D
- Goals:
  - Generate a fresh randomness for next epoch
  - Adversary cannot control, predict or pre-compute H(.)
- Solution:
  - Each final committee member pick a random $R_i$
  - Include $H(R_i)$ in the final block
  - Broadcast $R_i$ with final block
  - Each user takes an XOR of any C/2+1 random string $R_i$ receives

# Evaluation



Committee Formation time
Consensus Time
Blocks
Final Scalability
Time (s)
No. of Block
900
800
700
600
500
400
300
200
100
0
18
16
14
12
10
8
6
4
2
0
100
200
400
800
1600
Network size (number of nodes)

# Conclusion

- IPFS can be used to:
  - Deliver contents to websites
  - Globally store files with authomatic versioning and back up
  - Facilitate secure file sharing and encrypted communication

- Elastico: Computationally scalable protocol
  - By sharding securely
  - More computation power, higher transaction rate

# Discussion

1. What are the challenges of IPFS and blockchain integration?
2. Are the IPFS sub-protocols separated and defined logically?
3. What will happen to the scalability of the storage by use of sharding protocol?
4. How could sharding protocol be applied to the permissioned blockchains?