Bitcoin and Nakamoto Consensus

Distributed Systems

Nikita Borisov

Topics for Today

- Bitcoin
  - Consensus approach
  - Transaction broadcast

Bitcoin

- Implement a distributed, replicated state machine that maintains an *account ledger (= bank)*
- Scale to thousands of replicas distributed across the world
- Allow old replicas to fail, new replicas to join seamlessly
- Withstand various types of attacks

Approaches that don’t work

- Totally ordered multicast (e.g., ISIS)
  - Quadratic communication overhead
  - Do not know who all replicas are a priori

- Leader election (e.g., Bully)
  - Quadratic communication overhead
  - Do not know who all replicas are a priori
  - *Nodes with highest IDs are leaders =>*
    - Bottleneck
    - Security

Lottery Leader Election

- Every node chooses a random number
- Leader = closest to 0

Hash Functions

- Cryptographic hash function: H(x) -> { 0, 1, …, 2^{256}-1}
- Hard to *invert:*
  - Given y, find x such that H(x) = y
- E.g., SHA256, SHA3, …

- Every node picks random number x and computes H(x)
- Node with H(x) closest to 0 wins

Using a seed

- Every node picks x, computes H(seed || x)
  - Closest to 0 wins

What to use as a seed?

- Hash of:
  - Previous log
  - Node identifier
  - New messages to add to log
- Two remaining problems:
  - How to find closest to 0?
  - How to prevent nodes from trying multiple random numbers?

Iterated Hashing / Proof of work

find x s.t. $H(seed||x) < T$, $T = 2^{256}$

- Repeat:
  - Pick random x, compute y = H(seed || x)
  - If y < T, you win!
- Set threshold T so that on average, one winner every few minutes
- E.g.:
  - 1000 nodes
  - 10^12 hash/second
  - Target interval: 10 minutes
  - T = ?

$T / |\text{Domain}(H)|$

- Given a *solution*, x such that H(seed || x) < T, anyone can *verify* the solution in constant time (microseconds)

Bitcoin Summary

00000...f3ac

To commit a transaction to the log, a server must solve a puzzle:

← solution

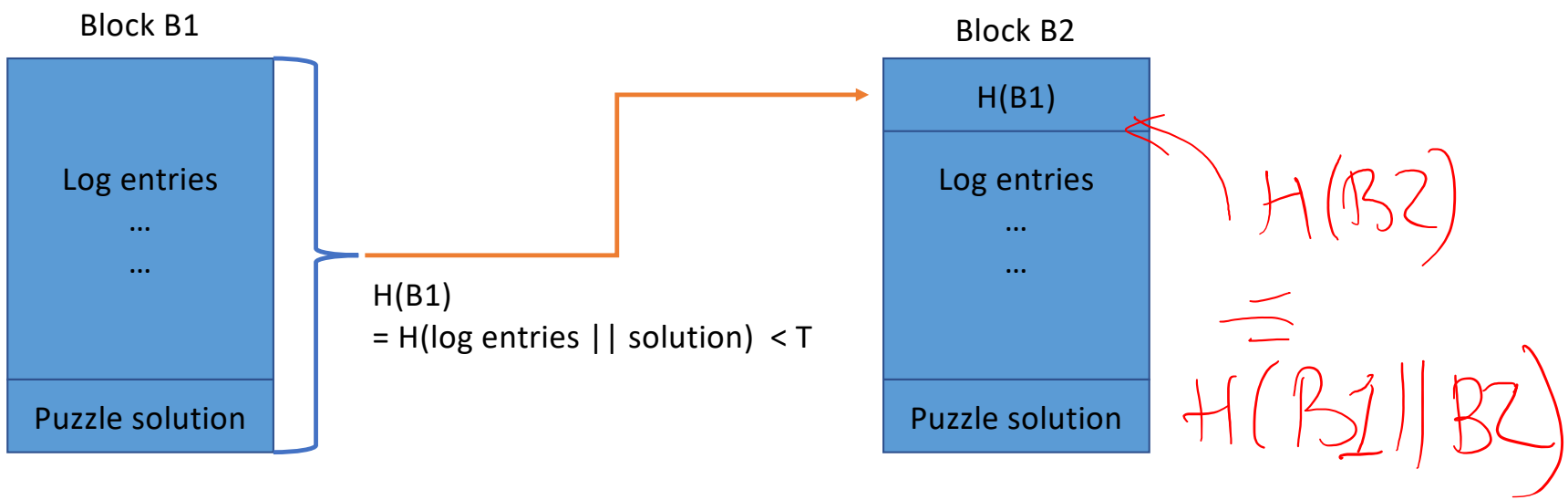
H(previous log | new tx | random input) < T

Any random input satisfies this with probability of $T/2^{256}$ so smaller T => more attempts needed to solve puzzle

Any node can *verify* the puzzle solution in one step

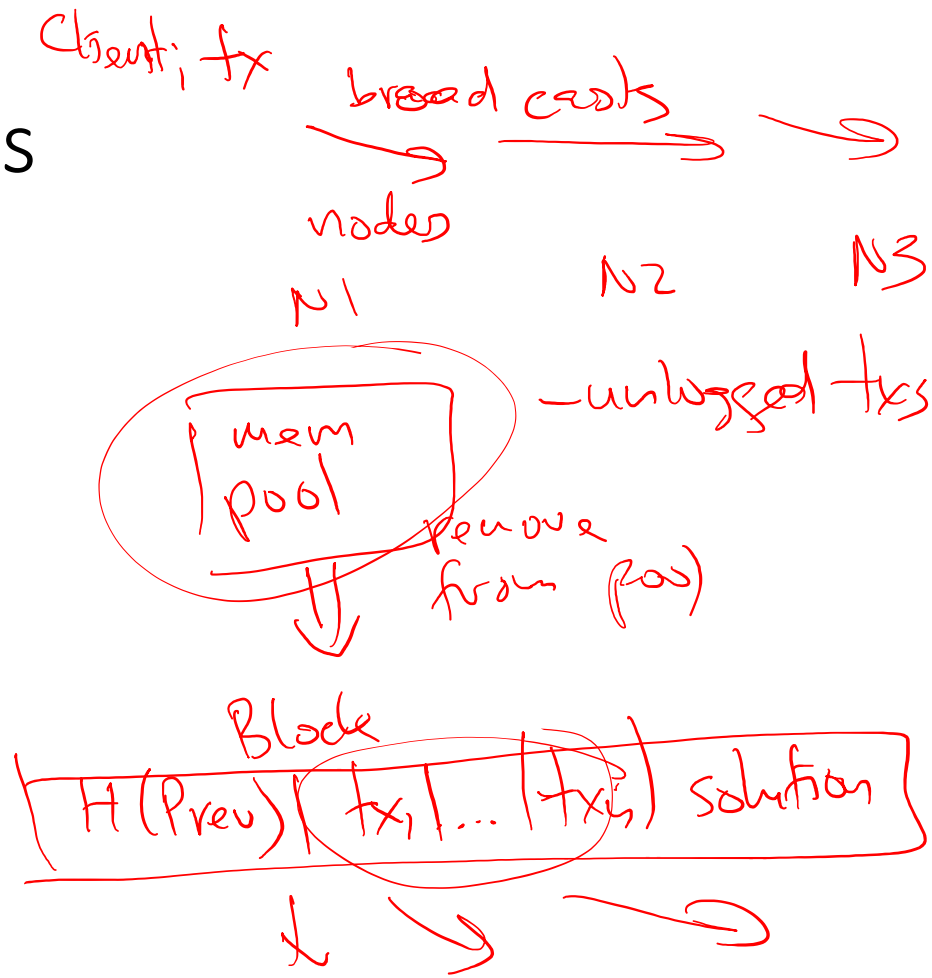
Block

Append Entries ([e1, e2, e3, ..., ek])



Bitcoin Node Operations

1. Clients broadcast transactions, servers collect unlogged transactions into mempool
2. Servers create candidate blocks, try to solve puzzle
3. If puzzle solved, add to log & broadcast block
4. If receive broadcast, verify solution & add to log
5. Restart step 2 with new log



Consensus Rules

A broadcast block is accepted into the log if it satisfies *consensus rules*

- Puzzle solution correct
- Block contents well-formed
- Transactions are authorized (digital signature of account owner)
- Balances don't go below 0

UTXO – unspent transaction output

Account ≡ public key

$\boxed{K^{pub}_{nikita}, \quad K^{sec}_{nikita}}$

auth sign (M, K^{sec}_{nikita})

$\Rightarrow$ sig

Verify $(M, sig, K^{pub}_{nikita}) = Y/N$

Double-spending

Account balance: $1000

TX1: Pay landlord $800

TX2: Buy TV for $750

At most one of these is valid

Forks

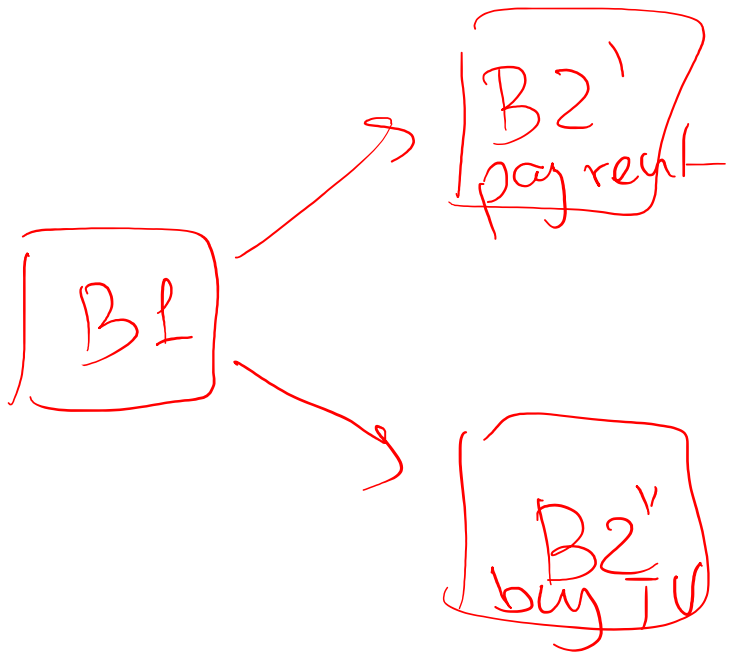
Two different blocks extending a single log == fork

Server solves puzzle before hearing another solution

- Reduce chance by making block time large, broadcast efficient

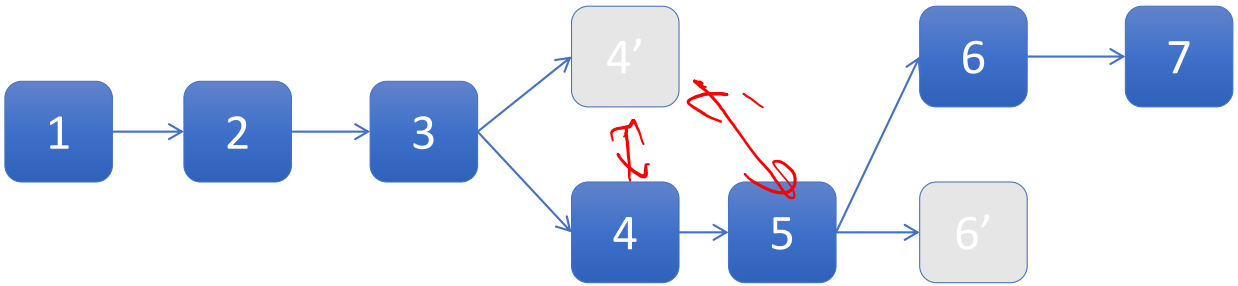
Server *chooses* to use old log to rearrange transactions

- Longest chain rule, incentives

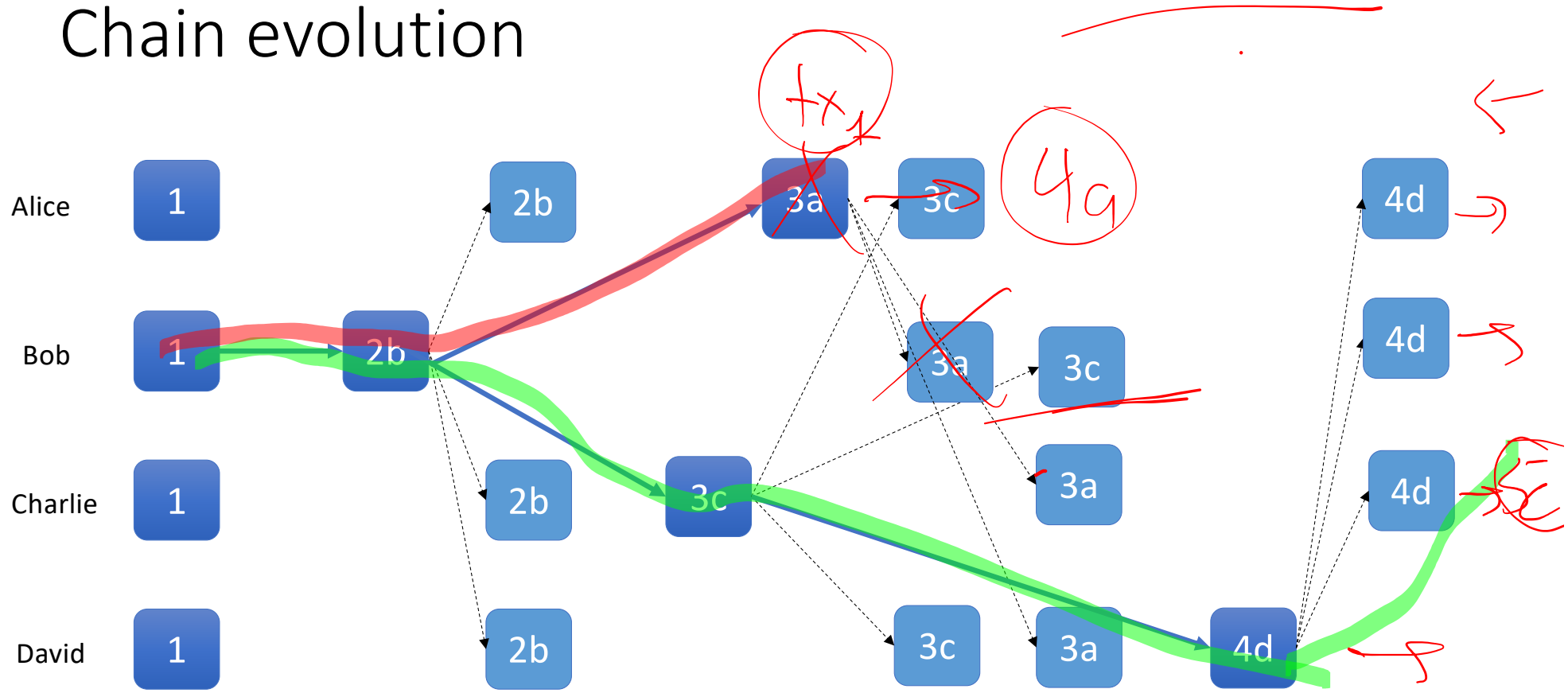


Chaining

- Each block's puzzle depends on the previous one
  - L_n -> L_{n-1} -> … -> L_1 -> L_0
  - To add m blocks, must solve m puzzles
- Longest chain wins



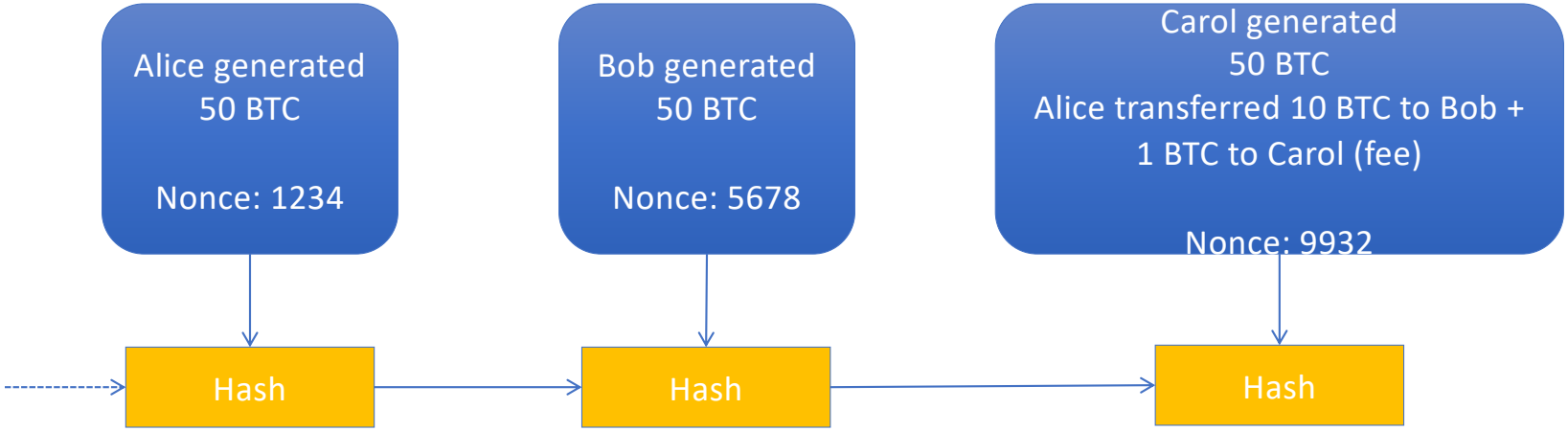
Chain evolution



Incentives for Logging

- Security better if more people participated in logging
- Incentivize users to log *others'* transactions
  - Transaction fees: pay me x% to log your data
  - Mining reward: each block *creates* bitcoins
    - Replace "Alice minted x" entries with "Alice logged line L_n"
- Payment protocol:
  - Alice->Bob: here's coin x
  - *Broadcast* to everyone: Alice transfers x to Bob
  - Bob: wait until transfer appears in a new log line
    - Optionally wait until a few more lines follow it

Putting it all together

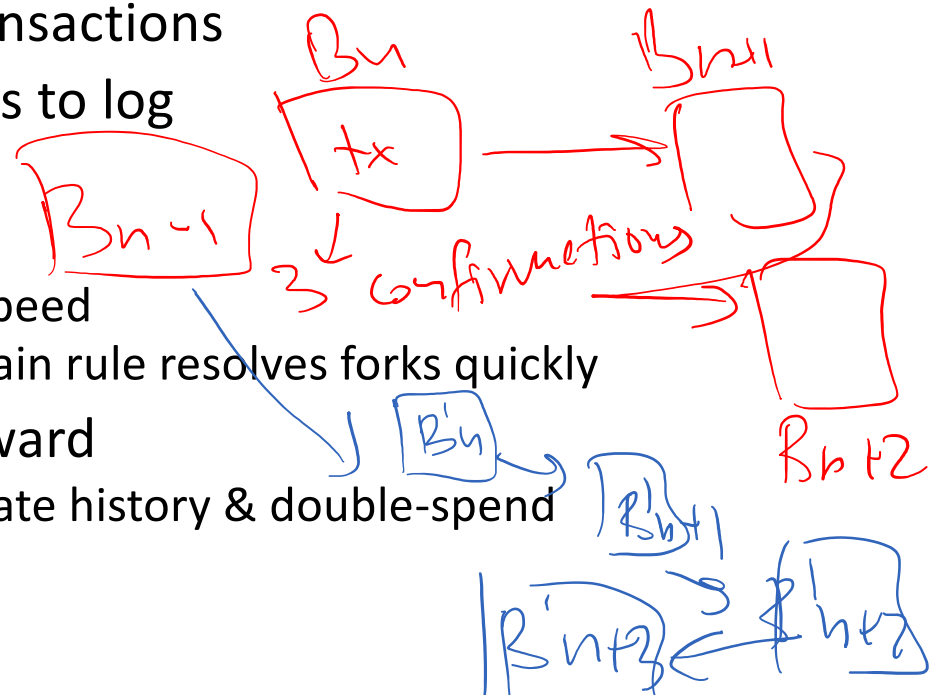


Account	Balance
Alice	39 BTC
Bob	60 BTC
Carol	51 BTC

Bitcoin Recap

def # of confirmations of tx = # of blocks after tx has been logged

- Clients broadcast transactions
  - Transfers between accounts, authenticated using public key signatures
- Miners create candidate blocks of transactions
- Miners solve puzzles to append blocks to log
  - Hard to solve, easy to verify
- Forks can happen, but …
  - Unlikely if broadcast speed << logging speed
  - Blocks / puzzles are chained, longest chain rule resolves forks quickly
- Mining incentivized by fees, block reward
  - Many miners => harder to create alternate history & double-spend



Logging Speed

- How to set T?
  - Too short: wasted effort due to broadcast delays & chain splits
  - Too long: slows down transactions
- Periodically adjust difficulty T such that one block gets added every 10 minutes
  - Determined algorithmically based on timestamps of previous log entries
- Current difficulty
  - $9 * 10^{22} =\sim 2^{76}$ hashes to win
- Large number of participants: hard to revise history!

Bitcoin broadcast

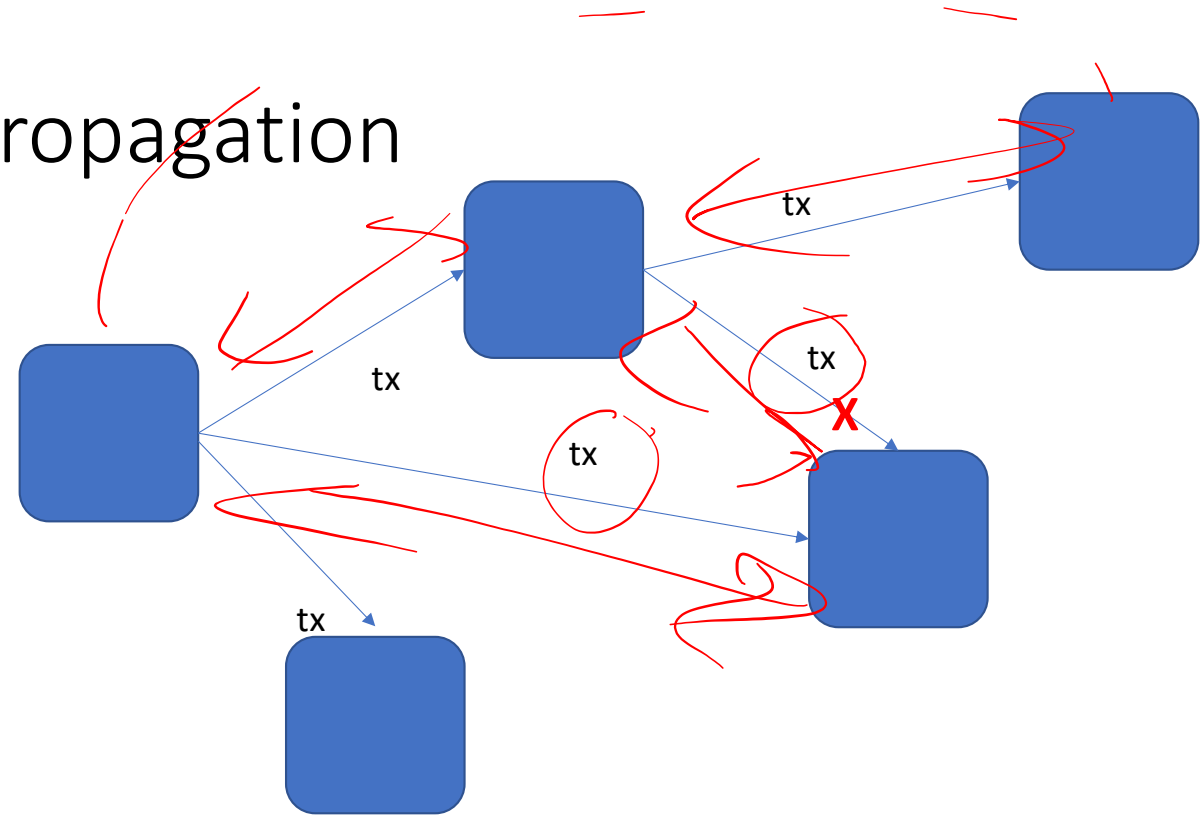
- Need to broadcast:
  - Transactions to all nodes, so they can be included in a block
  - New blocks to all nodes, so that they can switch to longest chain

- Why not R-multicast?
  - Have to send O(N) messages
  - Have to *know* which nodes to send to

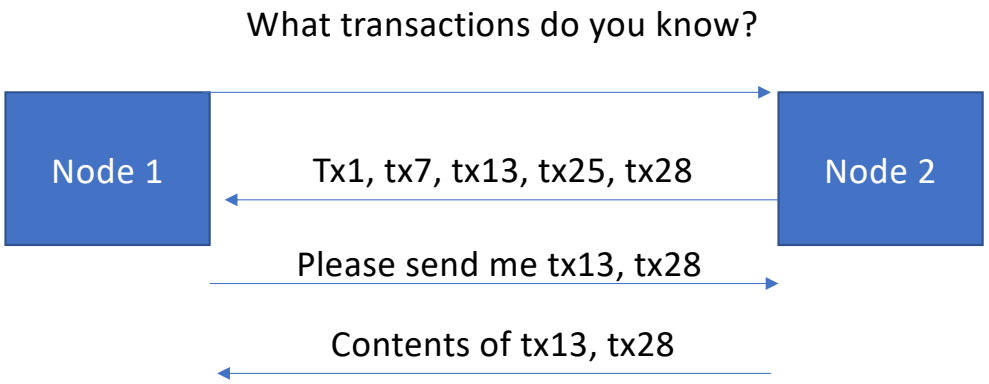
Gossip / Viral propagation

- Each node connects to a small set of *neighbors*
  - 10–100
- Nodes propagate transactions and blocks to neighbors
- Push method: when you hear a new tx/block, resend them to all (some) of your neighbors (flooding)
- Pull method: periodically poll neighbors for list of blocks/tx's, then request any you are missing

Push propagation

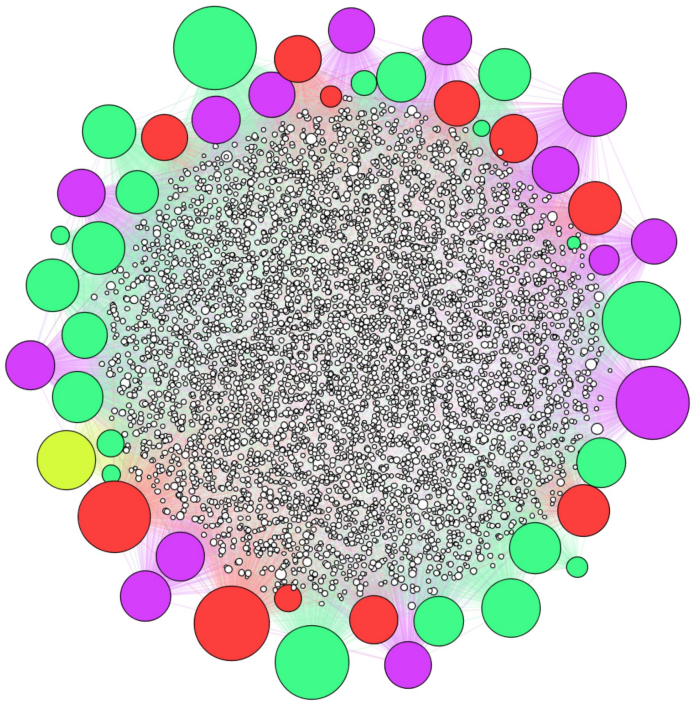


Pull propagation

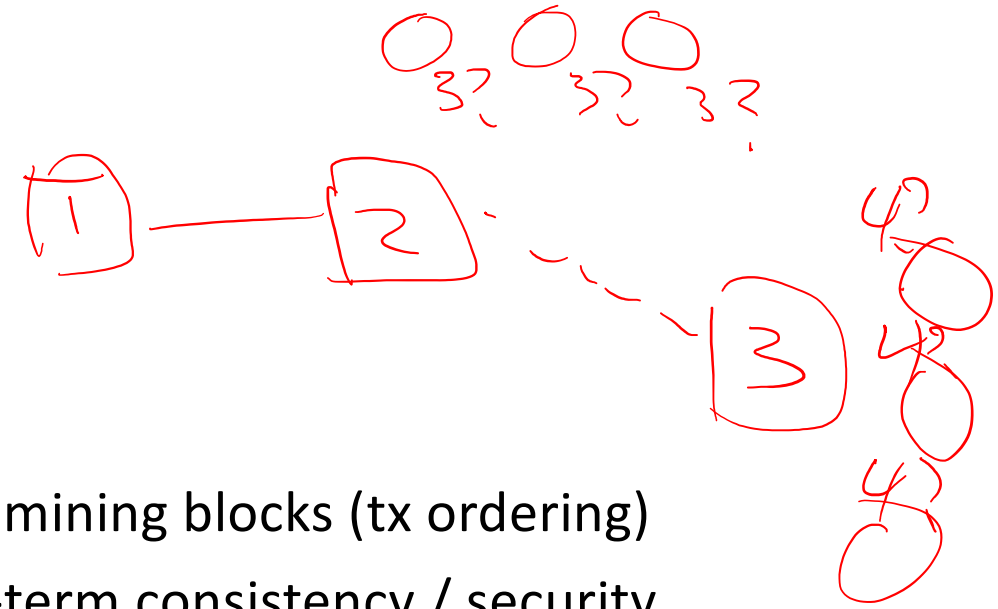


Maintaining Neighbors

- A *seed* service
  - Gives out a list of random or well-connected nodes
  - E.g., seed.bitnodes.io
- Neighbor discovery
  - Ask neighbors about *their* neighbors
  - Randomly connect to some of them



Bitcoin summary



Foundations:

- Unreliable broadcast using gossip
- Probabilistic “leader” election for mining blocks (tx ordering)
- Longest chain rule to ensure long-term consistency / security

Compared with Paxos/Raft:

- Scales to thousands of participants, dynamic groups
- Tens of minutes to successfully log a transaction (vs. milliseconds)

C → L → F → L → L
AE AER CONFIRM

Other Bitcoin Topics

- Mining pools to distribute winnings
- Security analysis, selfish-mining strategies, “incentive compatibility”
- Network design, gossip improvements, logging speed
- Block size, block propagation, block compression
- Private versions: Monero, Zcash, …
- Ethereum: build a full-scale virtual machine
  - Instead of executing transactions, execute programs
  - Programs maintain/update state, transfer money
  - Smart contracts control an account, receive commands