bitcoin

CS1699: Blockchain Technology and Cryptocurrency

6. Decentralization

Bill Laboon

ScroogeCoin Recap

- Using public-key cryptography (*à la* Goofycoin) and a blockchain validated by Scrooge (a centralized validator), we had a basic but functioning cryptocurrency
- HOWEVER - Scrooge was a weak point!
  - He generated all coins
  - He could punish users by not adding transactions to his blockchain
  - He could hold the entire system ransom by refusing to update

Decentralizing ScroogeCoin

- Can we obtain the same benefits of ScroogeCoin without a centralized party?
- Spoiler alert: **YES!** We will make a few simple attempts and see what other trade-offs are made to achieve it, as well as try to find problems with these simpler solutions

Centralization vs. Decentralization: The Very Idea

- There is always a tension between centralization and decentralization, not just in cryptocurrency
- Examples: Roman Empire vs Greek city-states, Apple vs Android ecosystems, mainframes vs personal computers
- Benefits and drawbacks - everything is a trade-off
- Most systems are a hybrid: ever try to set up your own SMTP server?

Bitcoin: Designed to Be Decentralized

"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."

–***Satoshi Nakomoto***, *"Bitcoin: A Peer-to-Peer Electronic Cash System" (the Bitcoin white paper)* https://bitcoin.org/bitcoin.pdf

How To Decentralize?

1. Who maintains the ledger of transactions?
2. Who has authority over which transactions are valid?
3. Who creates new bitcoins?
4. Who determines how the rules of the system change?
5. How do bitcoins acquire exchange value?

How Decentralized is Bitcoin?

- Different aspects = different levels of centralization, BUT possibility always exists for further decentralization
- *Bitcoin network itself* = Very decentralized - easy to spin up a fully-validating node
- *Mining* = Somewhat centralized - Open to anyone, but hard to do profitably and large economies of scale
- *Updating protocol/rules* = Very centralized in practice - Vast majority of nodes use Bitcoin Core. Anyone can write their own node software (or use an alternative such as BitcoinJ, Libbitcoin, BTCD, or Bitcoin Knots), but the VAST majority use Core
  - See https://bitnodes.earn.com/nodes/

Distributed Consensus

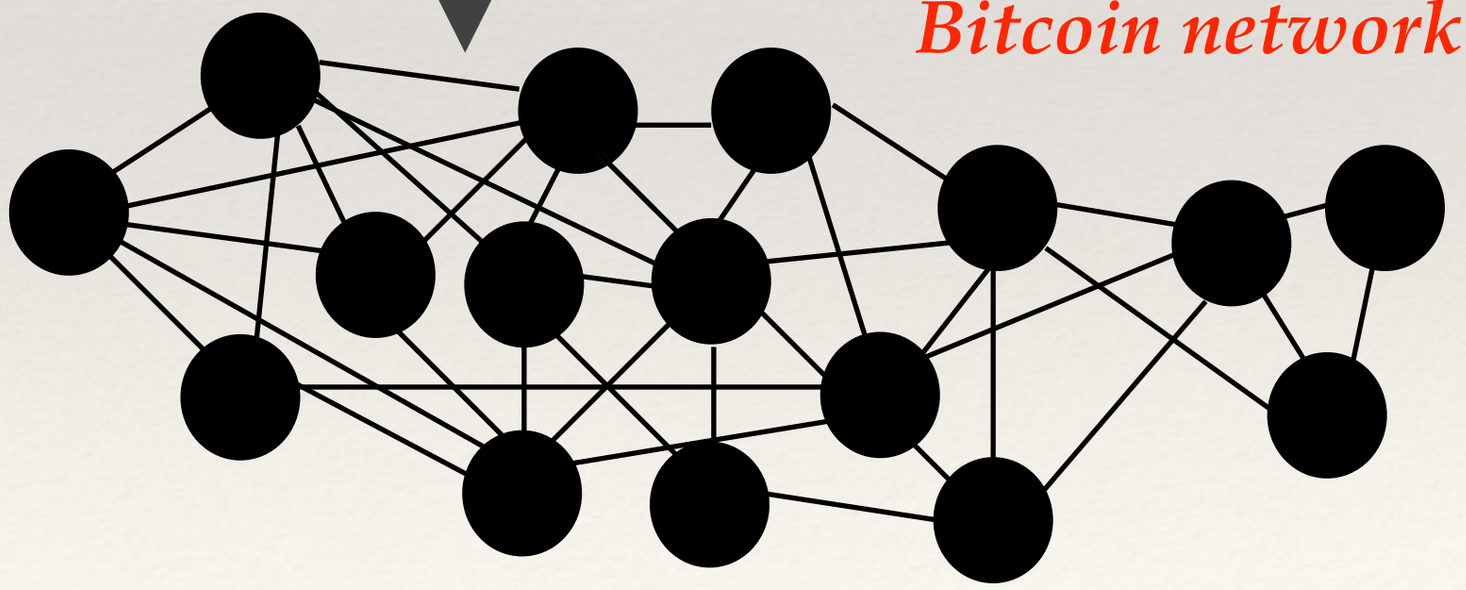
- How can a group of users come to a consensus on the "true" state of something (ledger, activity, status)?
- *Distributed consensus protocol*: There are *n* nodes that each have an input value. Some of these nodes are faulty or malicious. A distributed consensus protocl has the following properties:
  - It must terminate with all honest nodes in agreement on the value.
  - The value must have been generated by an honest node.

Transactions in Bitcoin

Alice

"I want to send Bob 1 bitcoin"

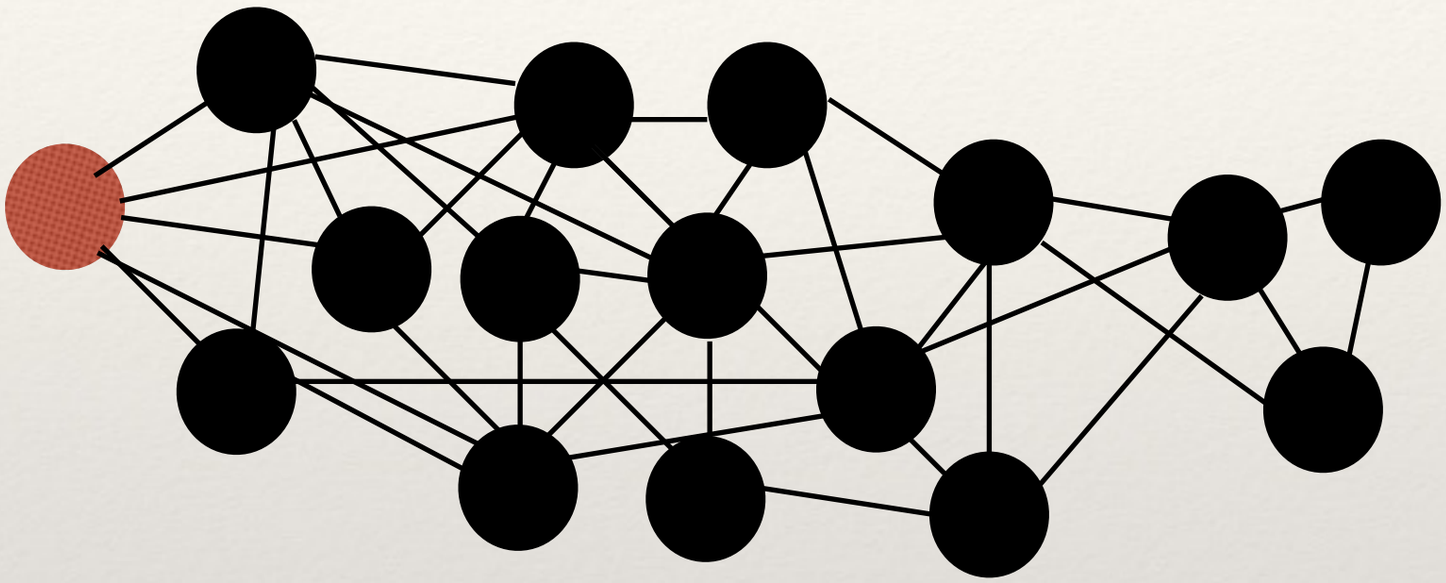
```
msg = "Pay pk_Bob 1 bitcoin"
sig = sign(msg, sk_Alice)
transmit(msg, sig)
```



Possible Consensus Mechanism

Bitcoin network

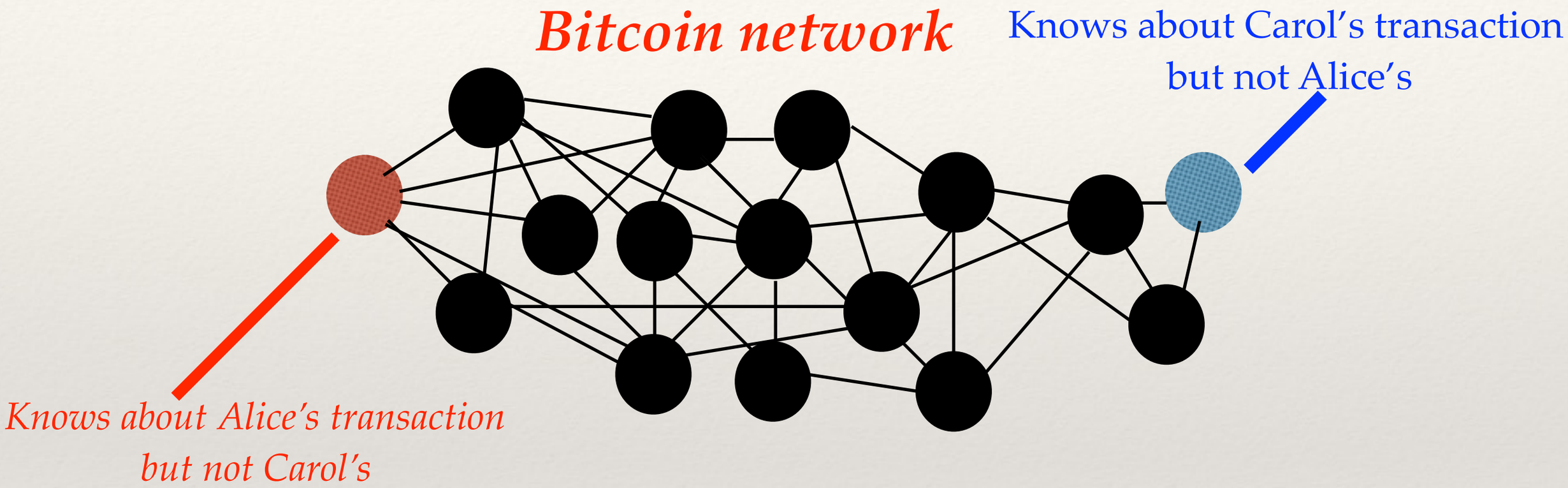
Initial node



Every node (circle) has:

1. The blockchain up to that point in time
2. A list of outstanding transactions which have been broadcast but are not yet in the blockchain (the *transaction pool* or *mempool*)

Possible Consensus Mechanism



Different nodes may have different transaction lists

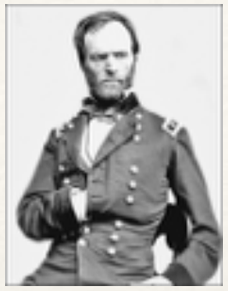
Every 10 minutes, nodes use some consensus mechanism (e.g. voting) to determine which transactions are put in blockchain

If a transaction is missed, no problem, wait for next block!

Problems with Naive Consensus Protocol

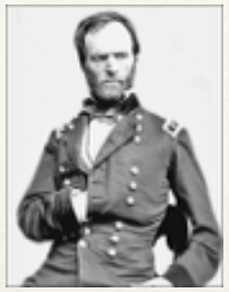
- No notion of universal time (who decides when it has been 10 minutes?)
- What consensus mechanism to use?
- Network is imperfect - some nodes will have high latency, faults in network, how many nodes actually exist?
- Malicious users could provide invalid/spam transactions

Byzantine Generals Problem



Lamport, Leslie et al. “The Byzantine Generals Problem.” ACM Transactions on Programming Languages and Systems, Volume 4, pp 382-401. https://www.microsoft.com/en-us/research/uploads/prod/2016/12/The-Byzantine-Generals-Problem.pdf

Solutions?



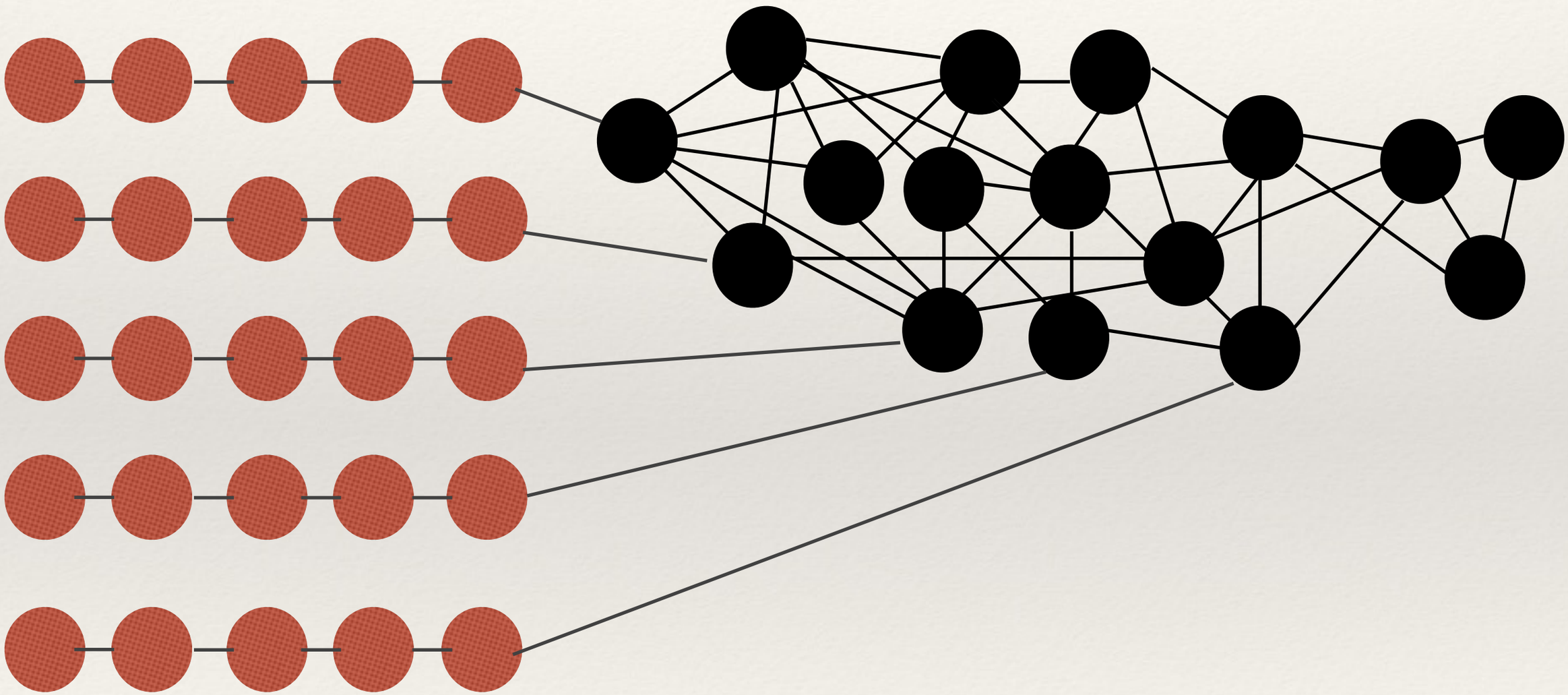
Distributed Systems and Time

- Segal's Law: *"A man with a watch knows what time it is. A man with two watches is never sure."*
- Lack of a centralized mechanism of time control (or even a supposedly-centralized mechanism like NTP) means that event-ordering is not strictly possible!
- Many common algorithms will not work when you don't/can't have a concept of "do x, *THEN* y"

Sybil Attacks

Eve's malicious nodes (25)

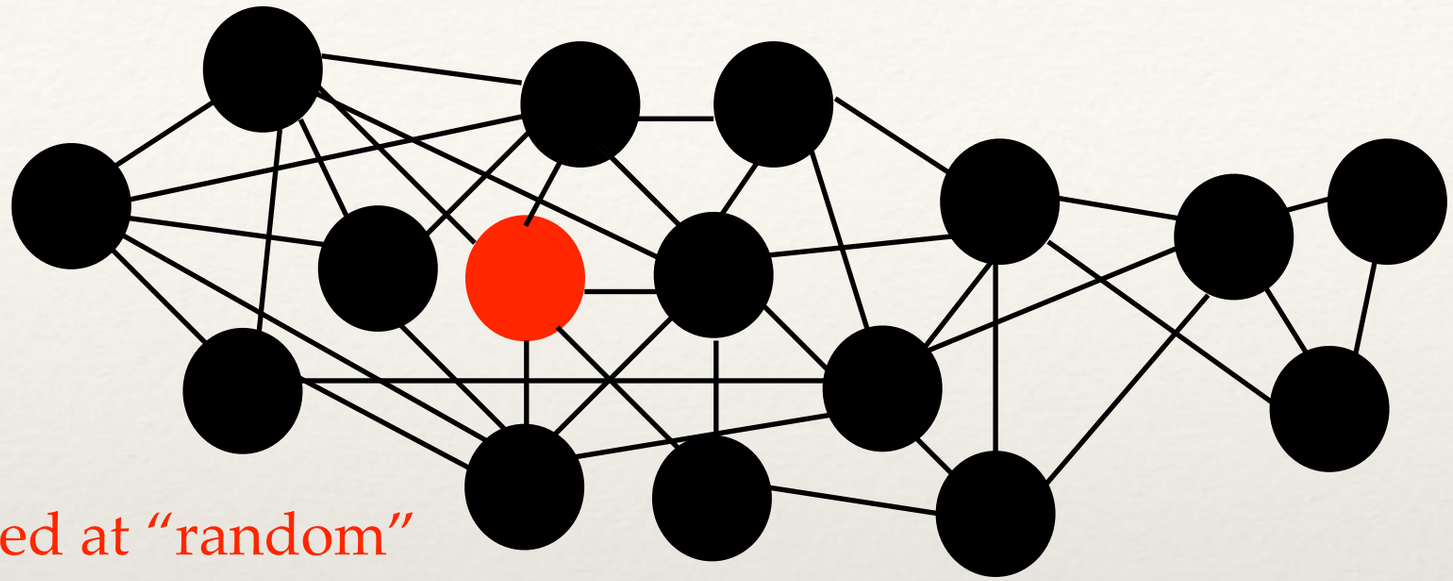
Honest nodes (16)



Simplified Bitcoin Consensus Algorithm

- New transactions broadcast to all nodes
- Each node collects new transactions into a block
- In each "round", a random node gets to broadcast its block *(note: assuming we can avoid Sybil attacks here!)*
- Other nodes accept block iff all transactions are valid
- Nodes express acceptance by including its hash in next block (lengthening the chain)

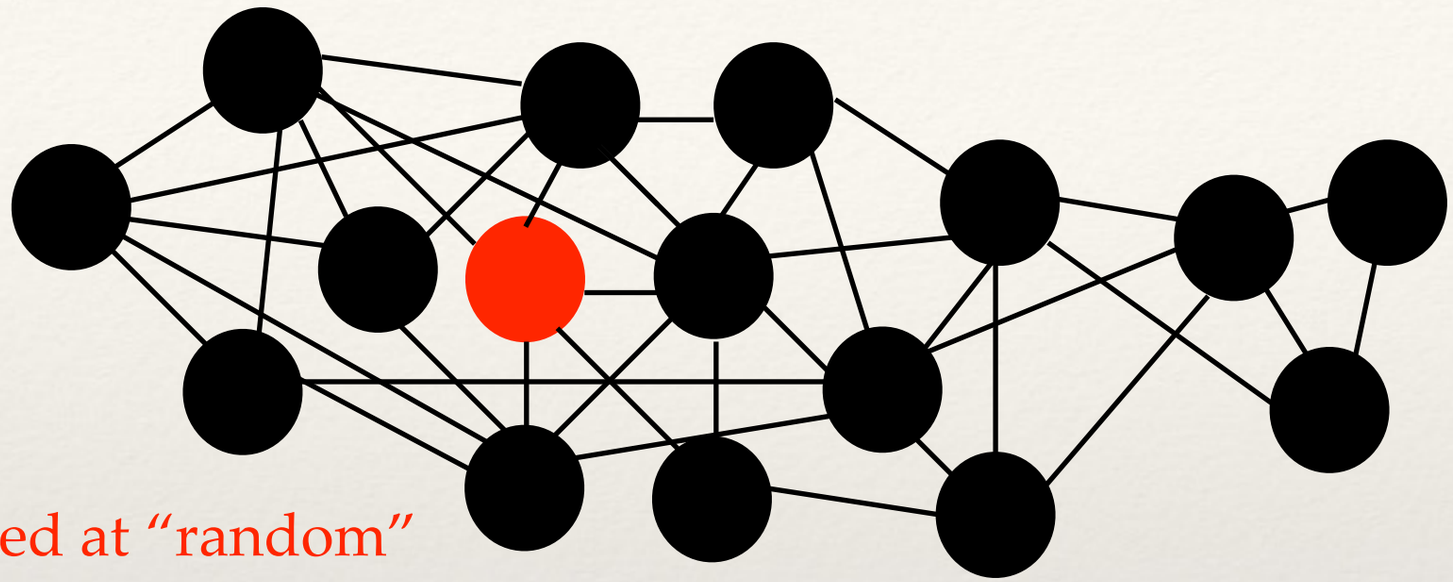
Simplified Consensus in Action



Red node picked at "random"

1. Recall all nodes have copy of blockchain
2. Red packages all valid transactions into block, adds to blockchain
3. Red broadcasts new blockchain
4. Other nodes add red's blockchain with new block and hash pointer

Denial of Service



Red node picked at “random”

1. Red packages all valid transactions (EXCEPT Bob’s) into block
2. Adds valid block to blockchain, which does not have Bob’s transactio
3. Red broadcasts new blockchain
4. Other nodes add red’s blockchain with new block and hash pointer

Sorry, Bob!

Double-Spend w/ Simplified Consensus

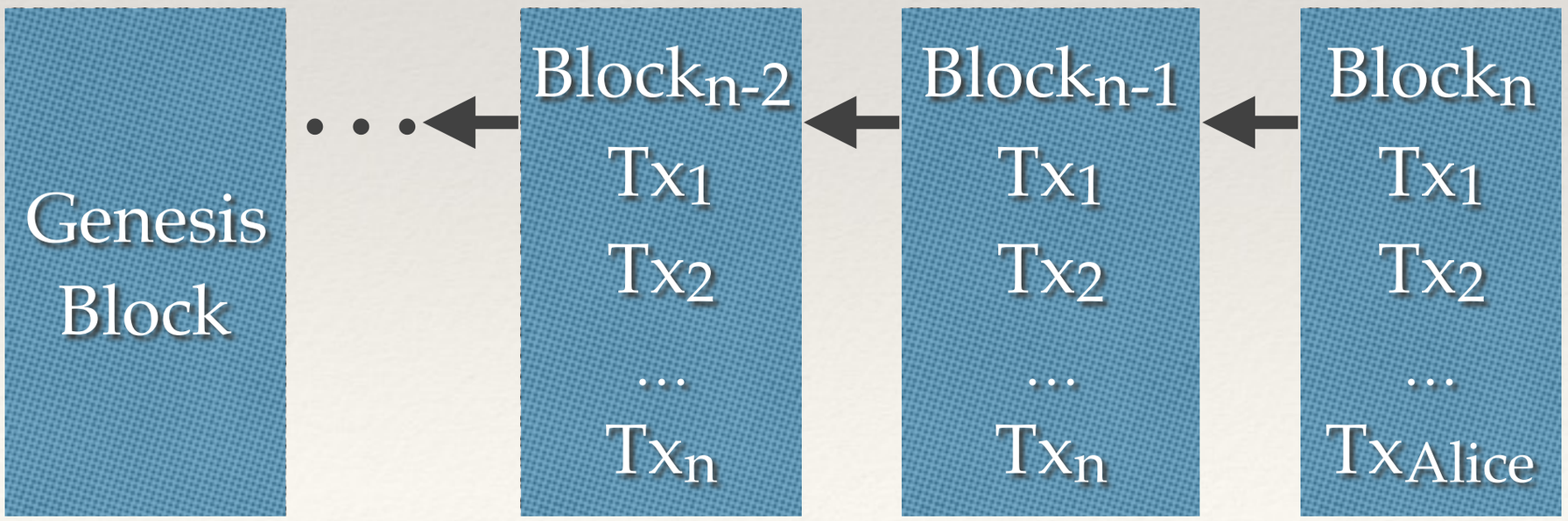
1. Alice purchases software from Bob for one bitcoin and generates a transaction to that effect
2. An honest node includes it in the blockchain
3. Bob sees that it has been confirmed, Alice downloads software
4. Alice node is selected for next block
5. Instead of building on block n, she builds on block n-1, ignoring the previous block and making her own block n where the bitcoin go to another address she controls
6. Transaction to Bob is useless - bitcoin in that block cannot be used by him in later blocks because it is not under his control

Double-Spend w/ Simplified Consensus

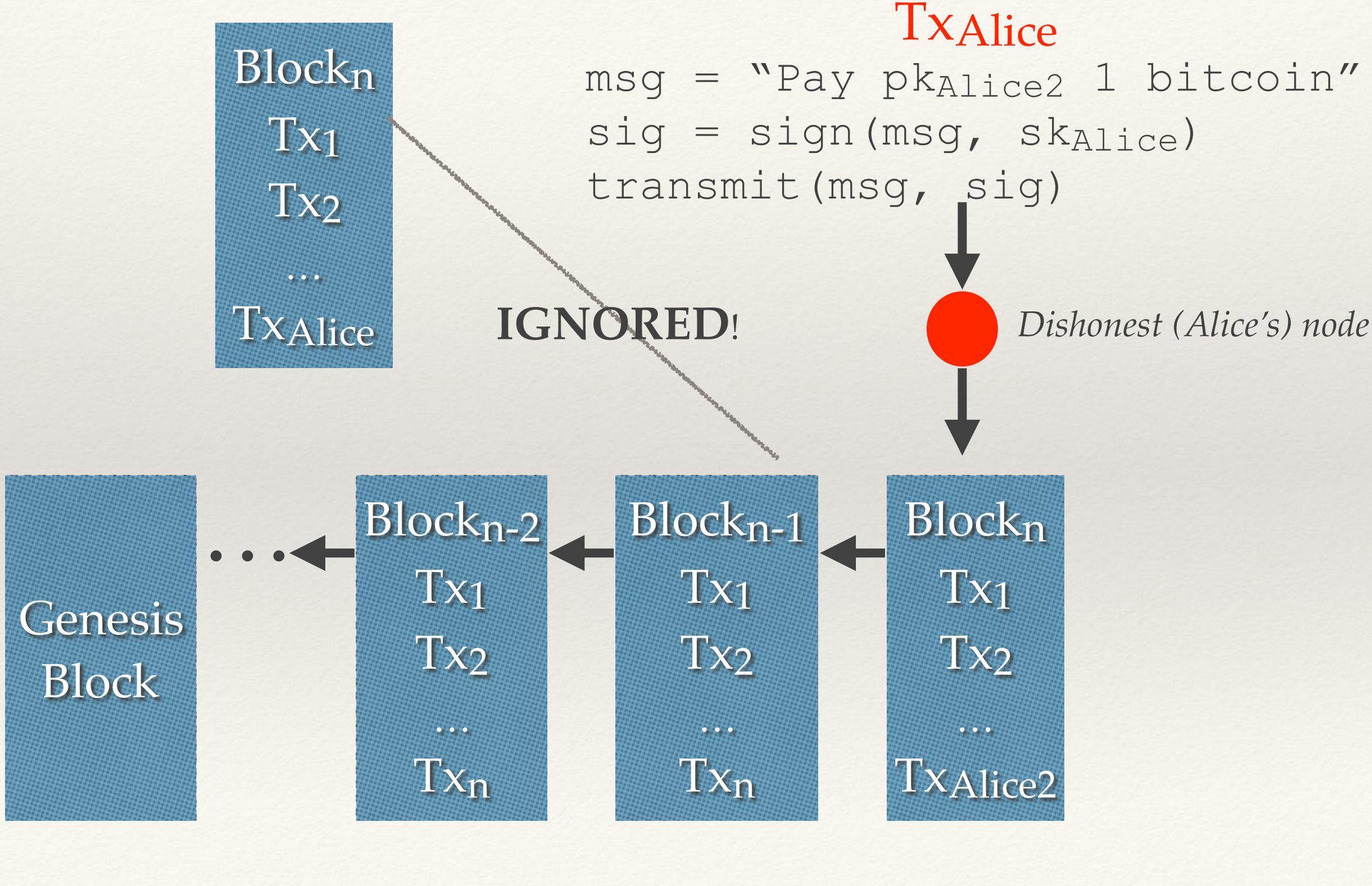
Tx_{Alice}

```
msg = "Pay pk_Bob 1 bitcoin"
sig = sign(msg, sk_Alice)
transmit(msg, sig)
```

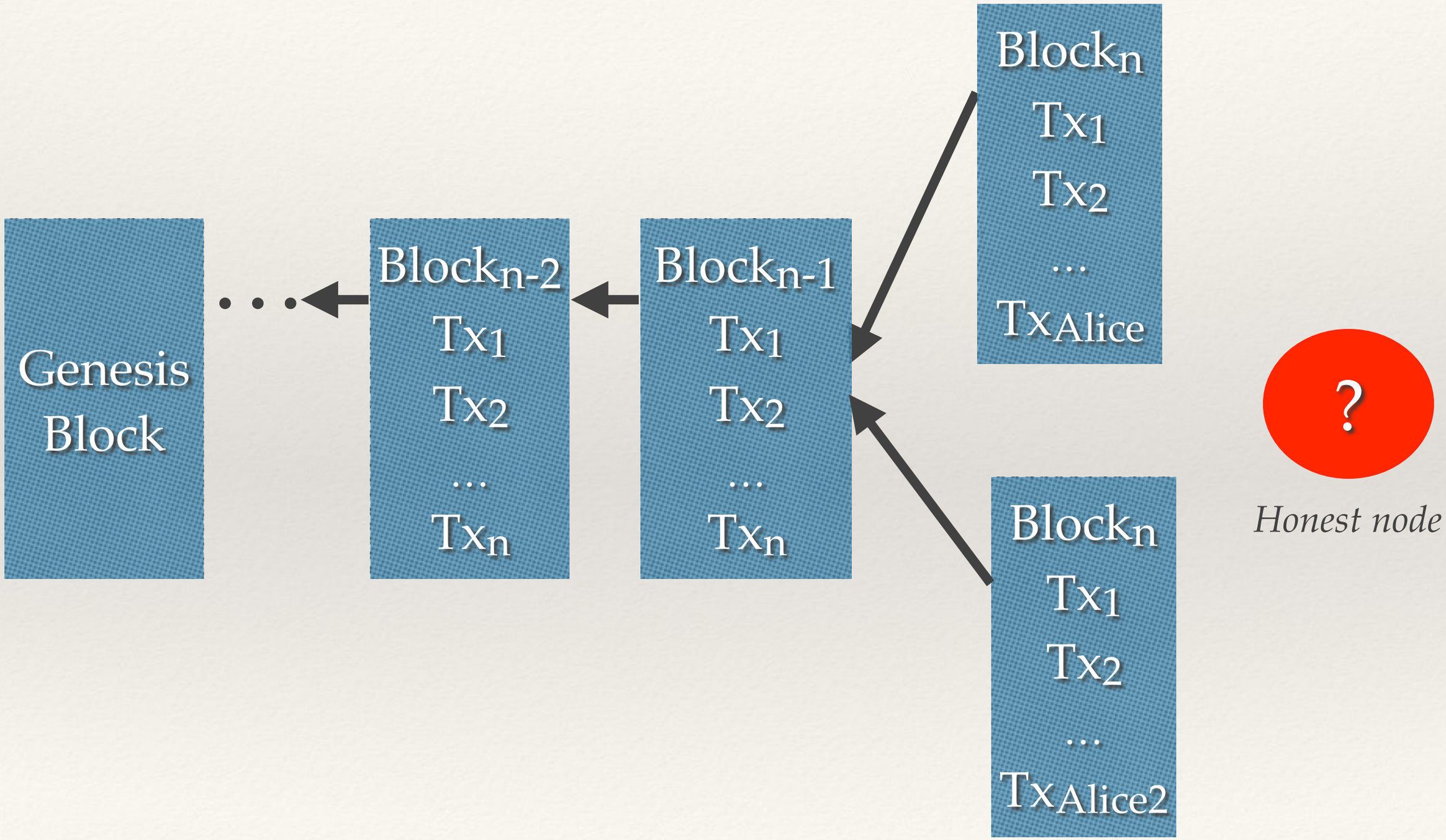
Selected (honest) node



Double-Spend w/ Simplified Consensus



Double-Spend w/ Simplified Consensus



Protecting Against Double-Spend with Simplified Consensus

- More blocks in blockchain since transaction, harder for someone to generate a different blockchain with different blocks
- General rule in Bitcoin: wait six blocks (heuristic)

What about Sybil Attacks?

- Two related assumptions: > 50% of nodes are honest, plus invulnerable to Sybil attacks
- Not true in practice… so how to avoid?
- Next lecture: incentives, proof of work, and mining