Centralization vs. decentralization is actually a central paradigm in many computer applications. For example: centralization reigns in social networking, but decentralization reigns in email. Let's see how it applies to Bitcoin and currencies more generally.

## Aspects of decentralization in Bitcoin

- **Peer-to-peer network**: Very low barrier to entry. Anyone can run a Bitcoin node. True P2P system.
- **Mining**: Open to anyone, but inevitable concentration of power (due to high capital cost) often seen as undesirable by the community. Slighly less decentralized than we might like.
- **Updates to software**: Core developers trusted by community, but have great power.

Comparing the centralized and decentralized models:

- From a **technical** perspective:
  - **Decentralized**: no single point of failure and fewer resource bottlenecks, as any node can fail and the system stays up. Nodes can come and go as they please.
  - **Centralized**: easier to compute statistics (e.g., even figuring out how many nodes there are is super difficult), easier to detect and control types of abuse.
- From a **socio-technical** perspective:
  - **Decentralized**: harder to standardize and change software once adopted. Bitcoin has this "hard-fork wish list" for changes that aren't backwards compatible, if there were ever a flag day at which everyone agreed to move to a new protocol. Once the system is standardized, becomes a great platform for innovation because they know it won't change in the future (similar to developing a Facebook app and praying that Facebook doesn't change the ToS).
- From an **economic** perspective:
  - **Decentralized**: tends to accrue hobbyist communities that spring up around decentralized systems. Willing to donate time, expense, expertise, computing power, all because they feel a sense of ownership in the decentralized system (similar to Linux, Wikipedia, etc.).
  - **Centralized**: economies of scale.
- From a **political** perspective:

  - **Decentralized**: harder to regular, but certainly not impossible.
  - **Centralized**: requires trust in the central authority.

## Distributed consensus

At a technical level, "distributed consensus" is the key challenge in creating a decentralized system. In other words: "How do we decentralize ScroogeCoin"?

The traditional motivation is **reliability** in distributed systems. Huge companies have tons of clusters, and when someone likes a photo, we might have 10-15 nodes in the back-end that know about the action. The system needs to ensure that *all* of the nodes eventually recognize this 'like', or *none* of the nodes recognize it.

**Definition** (Distributed Consensus)**.** *Assume there are n nodes or processes, and each of these nodes has some input value. A consensus protocol happens and the two requirements are:*

1. *That the protocol terminates and all correct nodes decide on the same value.*
2. *The value must have been proposed by some correct node.*

Recall that when Alice pays Bob, she broadcasts the transaction to *all* Bitcoin nodes. Bob's computer isn't even of particular interest in this sense. Alice sends in a hash that links together her receipt of the coin (from someone else previously) to her sending it to Bob now.

We want to reach consensus on:

1. Which transactions happened.
2. And in which order.

So, consensus might look like: *At any given time, all nodes have a sequence of* ***blocks*** *of transactions they've reached consensus on, and each node has a set of outstanding transactions its heard about (but have not reached consensus on).*

One way Bitcoin *might* implement consensus:

- Select *any* valid block, even if proposed by just one node.
- Add this block to the blockchain.
- Repeat.

This is actually a really hard technical problem for many reasons: nodes might crash, they might be malicious, there might be some latency or faults in the network, etc. There's also no notion of global time: not all nodes can agree to a common ordering of events based on timestamps, so you can't possibly design a system around that.

Because of this, there were many impossibility results proven around distributed consensus. However, they tell us more about the model than the problem, because they were proven to model distributed databases. Bitcoin violates many of their assumptions and, as a result, consensus in Bitcoin works better in practice than in theory.

### What does Bitcoin do differently?

- Introduce incentives: only possible because it's a currency.
- Embraces randomness: does away with the notion of a specific end-point for consensus. The consensus protocol takes roughly an hour, but even after that time, you only have a probabilistic guarantee that your transaction made it into the consensus blockchain.

## Consensus without identity

Nodes don't have a persistent, long-term identity. Why?

1. There's no central authority that can give identities to nodes and verify that they're not just creating new identities at-will (this is known as a *Sybil attack*).
2. Pseudonymity is a *goal* of Bitcoin.

Long-terms identities would be very useful. For example, protocols could assume that they have access to node identities, and we would be able to identity adversaries and malevolent nodes.

Instead, we often make a weaker assumption: *that we have the ability to pick a random node in the system.*

### Implicit consensus

In each round, a random node is picked. This node proposes the next block in the chain. Other nodes implicitly accept or reject the block:

- Accept by appending the proposed block to the blockchain.
- Reject by starting from the block just before.

This gives us the **consensus algorithm**:

1. New transactions are broadcast to all nodes.
2. Each node collects new transactions into a block.
3. In each round, a random node gets to broadcast its block.
4. Other nodes accept the block if all transactions in it are valid (inspect, with valid signatures).
5. Nodes express their acceptance of the block by including its hash in the next block they create.

How could a malicious adversary try to subvert this process?

- Say Alice proposes the next block in the chain and tries to steal Bitcoin. Impossible because she can't sign transactions.
- Say Alice tries to remove any of Bob's transactions. Bob can just wait another block until an honest node proposes another block and his transaction will sneak in there.
- **Double-spending attack**: Alice attempts to spend a Bitcoin twice. Goes to some site, creates a transaction from her address to Bob's address, and broadcasts the transaction. An honest node listens to the transaction and includes it in their next block. Alice gets to propose the next block and ignores the previous block, instead including a transaction from herself to another address $A'$ controlled by her (i.e., she wants to send her money back to her). Will it succeed?
  - Depends on which transaction makes it into the consensus chain.
  - Honest nodes always try to extend the longest valid branch.
  - These two transactions look identical, even though one is from Alice to Bob and the other is from Alice to Alice. Nodes have no idea which is "moral".
  - There's some chance that the next block opts for Alice's self-spend than for her Alice-to-Bob transaction. And then the next node is much more likely to keep extending that path, as it is now the largest valid chain.
  - Let's look at this from Bob's point-of-view.
    - Immediately, Bob hears about Alice's spend. Before hearing any confirmation, he sends her the goods (this is a "0 confirmation" transaction).
    - He could also wait for 1 confirmation before approving. Or 2 confirmations. Or more. **The more confirmations your transactions get, the higher the probability that it ends up in the long-term consensus chain.**
    - In general, the double-spend probability decreases *exponentially* with the number of confirmations. Most heuristics use 6 confirmations (good tradeoff between the amount of time you'll have to wait and the guarantee that your transaction ends up in the blockchain).

To recap these attempts:

- Protection against invalid transactions is cryptographic, but enforced by consensus.
- Protection against double-spending is purely by consensus.
- You're never 100% sure that a transaction you're interested in is actually part of the consensus branch.

Note that a transaction contains:

- Alice's signature.
- Bob's public key address.
- The hash of the transaction in which Alice received the coin she's sending to Bob (this transaction needs to be included in a previous block in the consensus chain).

## Incentives and proof of work

Recall that Bitcoin's decentralization is partly a matter of incentive engineering. Assuming honesty is problematic–can we give nodes **incentives** for behaving honestly?

Can we penalize a node that attempts to double-spend? No, because nodes don't have identities.

Instead, can we reward nodes that created the blocks that made it to the long-term consensus blockchain? Yes–by rewarding them with Bitcoins.

Everything thus far was just an abstract algorithm. Now, consider that we've actually been implementing a currency with value.

### Incentive 1: Block Reward

According to the rules of Bitcoin, the creator of a block gets to include a special coin-creation transaction in the block **and** choose the recipient address. The current reward is 25 BTC but halves every four years.

Why is this useful? How does this incentivize honest behavior? **The block creator gets to "collect" the reward *only* if the blocks ends up on long-term consensus branch–otherwise, this coin-creation transaction won't be considered valid in the long-term.**

This is an incredibly subtle but important mechanism.

#### Halving Mechanism

The halving mechanism leads to a finite supply of 21 million Bitcoins (based on geometric series). This is a final number for how many Bitcoins there can ever be, as this is the *only* way to create new Bitcoins.

### Incentive 2: Transaction Fees

The creator of a transaction (not a block, but a transaction) can choose to make the output value less than the input value. The remainder is a transaction fee that goes to the block creator. Note that this fee is *purely voluntary*, like a tip.

## Proof of work

To approximate selecting a random node, we select nodes in proportion to a resource that no one can monopolize (we hope). This could be in proportion to computing power (**proof-of-work**) or in proportion to ownership of the currency (proof-of-stake, not used in Bitcoin).

With proof-of-work, we allow nodes to compete with each other by using their computing power. Alternatively, we make it moderately hard to create new identities because they will need to prove their work.

The actual proof-of-work model uses **hash puzzles**:

**Definition** (Hash Puzzle)**.** *In order to create a block, the node must find a nonce such that the hash of the nonce, the previous hash, and the list of transactions that comprise that block ($H(nonce||prev\_hash||tx||...||tx)$) is very small. In other words, the node needs to find an output value that's small out of the total output space.*

*If the hash function is secure, **the only way to succeed is to try enough nonces until you get lucky**.*

Fundamentally, this is the computational problem that the node is required to solve. At all times, nodes are independently competing to solve these hash puzzles. Every once in a while, a node gets lucky and guesses a nonce; then, this node gets to propose the next block.

There are three essential properties here:

1. As of August 2014: you need to compute roughly $10^{20}$ hashes per block. Only some nodes bother to compete–these are known as **miners**.

2. This cost needs to be parameterizable. Nodes automatically re-calculate the target every two weeks, maintaining the invariant that the average amount of time used to crack a block globally is about 10-minutes. We can sum this up with a formula: $\mathbb{P}$[Alice wins next block] = (fraction of global hash power she controls).

3. It's trivial to verify that the node has found a nonce. As such, the nonce gets published as part of the block, and other miners simply verify that $H(...) < target$.

Our new key security assumption: attacks are infeasible if the majority of minors *weighted by hash power* follow the protocol (because we now have reasonable guarantees that the next block to be proposed will be by an honest node).

Miners need to decide if the costs of the hardware and electricity are less than that of the mining rewards. But this equation is complicated by:

- The ever-changing ratio of their own computing power vs. the global hash rate.

- The fact that they're rewarded in Bitcoin, but they're paying dollars, and so the transaction rate is important to consider.

- Some of these costs are fixed and others are variable.

## Putting it all together

- **Identities**: There are no real-world identities required to participate in Bitcoin. Any user can create any number of identities.
- **Transactions**: Broadcast over the network. Instructions to transfer one coin from some address to another.
- **P2P network**: Goal is to propagate new transactions to Bitcoin peer nodes.
- **blockchain & consensus**: If your transaction makes it into the blockchain, it has a lot of confirmation.
- **Hash puzzles & mining**: Miners are special types of nodes that opt in to this game of solving hash puzzles. They are rewarded in Bitcoin and are then allowed to propose new blocks for the chain.

When I say I own a certain amount of Bitcoin, I mean that the Bitcoin P2P network considers me to own a certain amount of Bitcoin.

## Bootstrapping

There's a tricky interplay between three things in Bitcoin: the *security of the blockchain*, the *value of the currency*, and the *health of the mining ecosystem*.

We need it to be the case that an adversary can't takeover the blockchain (i.e., the blockchain is secure). That will only be true if the mining ecosystem is healthy and made up of honest, protocol-following nodes. That will only be true if the currency is valuable. And that will only be true if the blockchain is secure.

In the beginning, none of these three things existed. There were no miners except Satoshi, and there was no value to the currency. There was even an insecure blockchain due to lack of miners. Since then, Bitcoin has bootstrapped itself. **Bitcoin went from having none of these properties to having (in some large measure) all of them.**

## What can a "51% attacker" do?

A good way to understand consensus is to say: "What would happen if consensus failed and there were some attack that controlled 51% or more of the mining power?"

- *Could this attacker steal coins from an existing address?* They could pretend there's some valid block (even if the crypto denied it) in which it steals Bitcoin. But the other honest nodes would reject this block. There would be a *fork* in the chain. Then, when the attacker earns their Bitcoin reward, they wouldn't be able to spend their Bitcoin with others. **Subverting consensus is not enough. You have to subvert consensus to steal Bitcoins.**

- *Could this attacker suppress transactions?* The attacker can refuse to create new blocks that contain transactions containing Carol's addresses. However, the attacker cannot prevent these transactions from being broadcasted to the rest of the network and reach the majority of nodes. *The attack will be very clear, if it happens.*
- *Could this attacker change the block reward?* Not possible. The attacker doesn't control the copies of the Bitcoin software that the honest nodes are running.
- *Could this attacker destroy confidence in Bitcoin?* If there were a variety of double-spend attempts and lots of odd behavior, then individuals would decide that Bitcoin were no longer behaving in a decentralized way. People would lose confidence and the exchange rate would plummet. **This is a real practical threat to Bitcoin.**