The Meaning of Decentralization

Vitalik Buterin Feb 6, 2017

"Decentralization" is one of the words that is used in the cryptoeconomics space the most frequently, and is often even viewed as a blockchain's entire raison d'être, but it is also one of the words that is perhaps defined the most poorly. Thousands of hours of research, and billions of dollars of hashpower, have been spent for the sole purpose of attempting to achieve decentralization, and to protect and improve it, and when discussions get rivalrous it is extremely common for proponents of one protocol (or protocol extension) to claim that the opposing proposals are "centralized" as the ultimate knockdown argument.

But there is often a lot of confusion as to what this word actually means. Consider, for example, the following completely unhelpful, but unfortunately all too common, diagram:

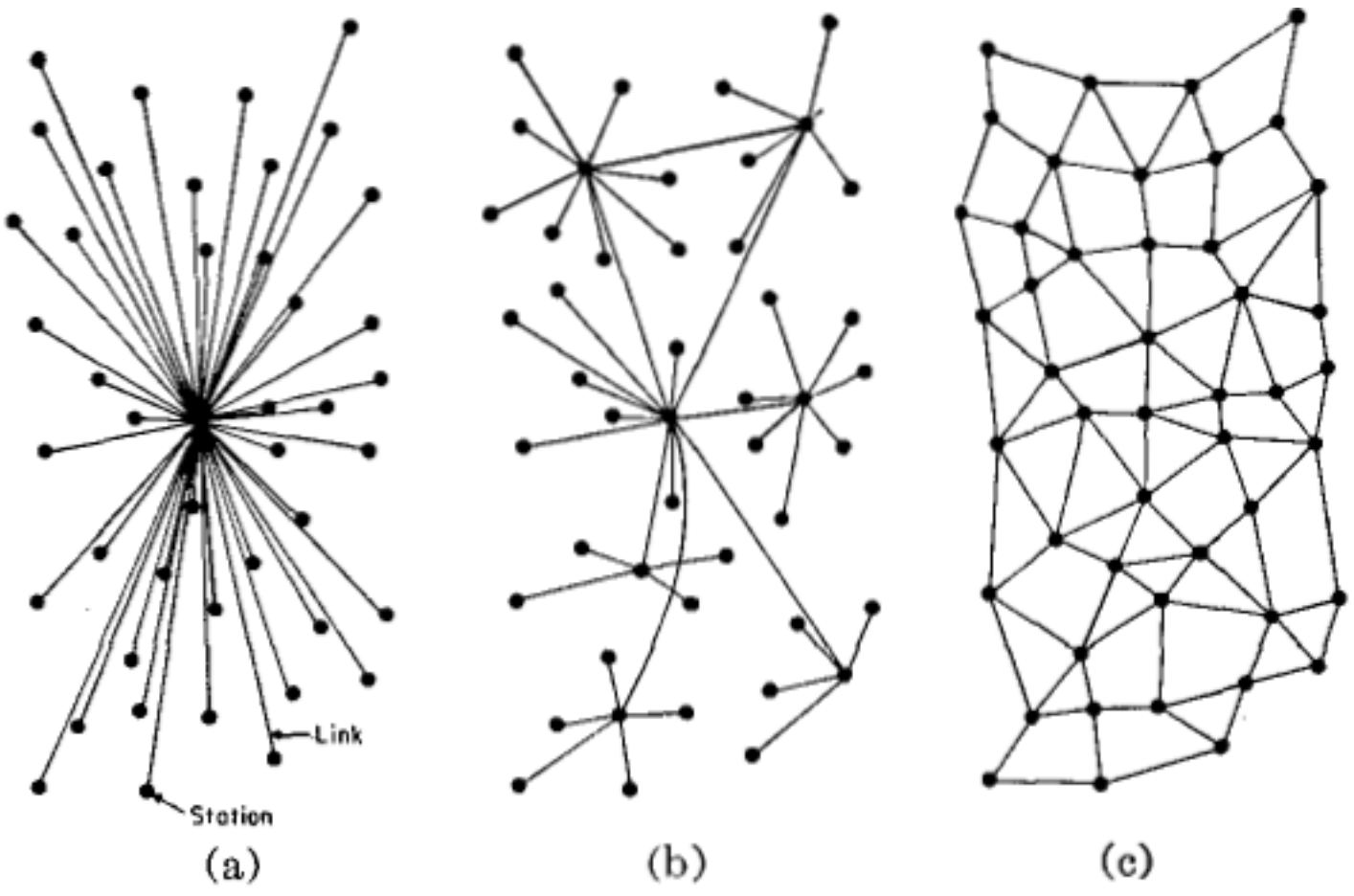


Fig. 1—(a) Centralized. (b) Decentralized. (c) Distributed networks.

Now, consider the two answers on Quora for "what is the difference

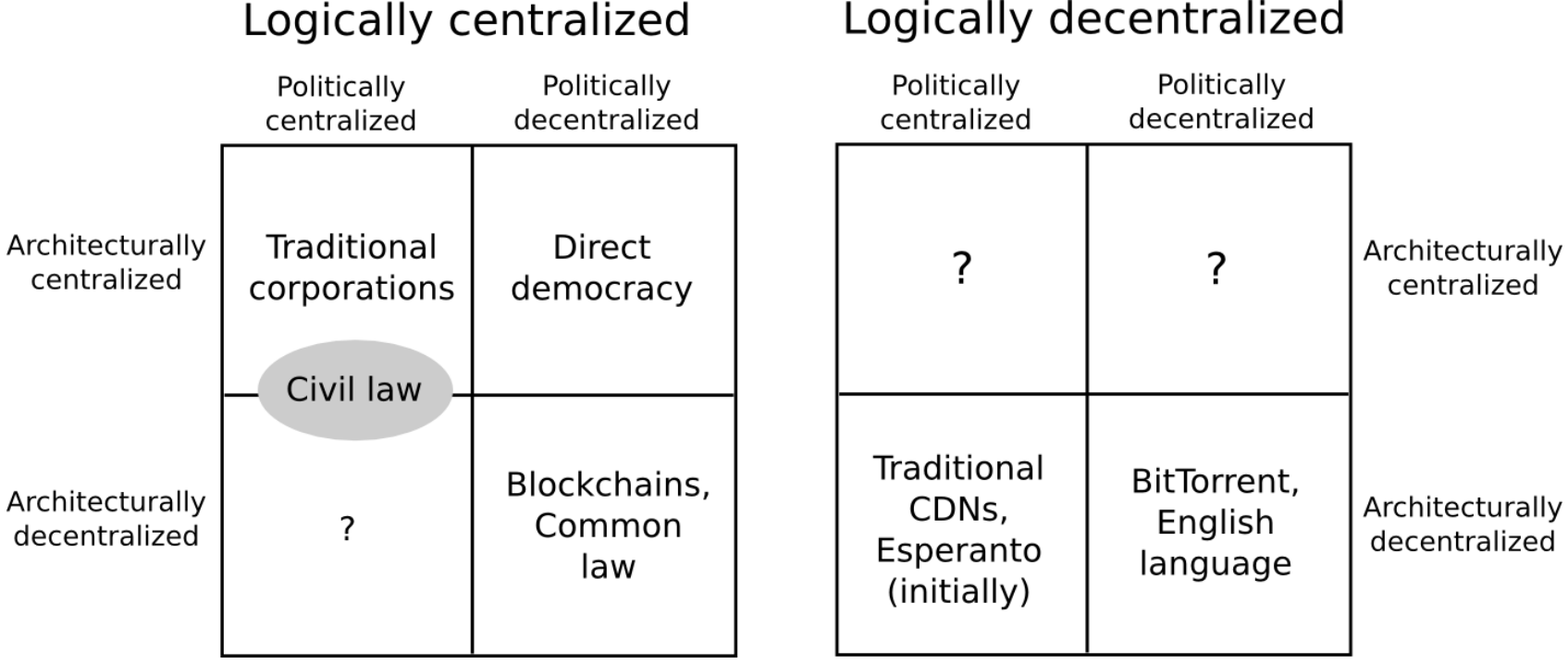
between distributed and decentralized". The first essentially parrots the above diagram, whereas the second makes the entirely different claim that "distributed means not all the processing of the transactions is done in the same place", whereas "decentralized means that not one single entity has control over all the processing". Meanwhile, the top answer on the Ethereum stack exchange gives a very similar diagram, but with the words "decentralized" and "distributed" switched places! Clearly, a clarification is in order.

Three types of Decentralization

When people talk about software decentralization, there are actually *three separate axes* of centralization/decentralization that they may be talking about. While in some cases it is difficult to see how you can have one without the other, in general they are quite independent of each other. The axes are as follows:

- **Architectural (de)centralization** — how many **physical computers** is a system made up of? How many of those computers can it tolerate breaking down at any single time?
- **Political (de)centralization** — how many **individuals or organizations** ultimately control the computers that the system is made up of?
- **Logical (de)centralization**— does the **interface and data structures** that the system presents and maintains look more like a single monolithic object, or an amorphous swarm? One simple heuristic is: if you cut the system in half, including both providers and users, will both halves continue to fully operate as independent units?

We can try to put these three dimensions into a chart:



Note that a lot of these placements are very rough and highly debatable. But let's try going through any of them:

- Traditional corporations are politically centralized (one CEO), architecturally centralized (one head office) and logically centralized (can't really split them in half)
- Civil law relies on a centralized law-making body, whereas common law is built up of precedent made by many individual judges. Civil law still has some architectural decentralization as there are many courts that nevertheless have large discretion, but common law have more of it. Both are logically centralized ("the law is the law").
- Languages are logically decentralized; the English spoken between Alice and Bob and the English spoken between Charlie and David do not need to agree at all. There is no centralized infrastructure required for a language to exist, and the rules of English grammar are not created or controlled by any one single person (whereas Esperanto was originally invented by Ludwig Zamenhof, though now it functions more like a living language that evolves incrementally with no authority)
- BitTorrent is logically decentralized similarly to how English is. Content delivery networks are similar, but are controlled by one single company.
- Blockchains are politically decentralized (no one controls them) and architecturally decentralized (no infrastructural central point of

failure) but they are logically centralized (there is one commonly agreed state and the system *behaves* like a single computer)

Many times when people talk about the virtues of a blockchain, they describe the convenience benefits of having "one central database"; that centralization is logical centralization, and it's a kind of centralization that is arguably in many cases good (though Juan Benet from IPFS would also push for logical decentralization wherever possible, because logically decentralized systems tend to be good at surviving network partitions, work well in regions of the world that have poor connectivity, etc; see also this article from Scuttlebot explicitly advocating logical decentralization).

Architectural centralization often leads to political centralization, though not necessarily—in a formal democracy, politicians meet and hold votes in some physical governance chamber, but the maintainers of this chamber do not end up deriving any substantial amount of power over decision-making as a result. In computerized systems, architectural but not political decentralization might happen if there is an online community which uses a centralized forum for convenience, but where there is a widely agreed social contract that if the owners of the forum act maliciously then everyone will move to a different forum (communities that are formed around rebellion against what they see as censorship in another forum likely have this property in practice).

Logical centralization makes architectural decentralization harder, but not impossible—see how decentralized consensus networks have already been proven to work, but are more difficult than maintaining BitTorrent. And logical centralization makes political decentralization harder—in logically centralized systems, it's harder to resolve contention by simply agreeing to "live and let live".

Three reasons for Decentralization

The next question is, why is decentralization useful in the first place? There are generally several arguments raised:

- **Fault tolerance**— decentralized systems are less likely to fail accidentally because they rely on many separate components that are not likely.
- **Attack resistance**— decentralized systems are more expensive to attack and destroy or manipulate because they lack sensitive central points that can be attacked at much lower cost than the economic size of the surrounding system.
- **Collusion resistance**—it is much harder for participants in decentralized systems to collude to act in ways that benefit them at the expense of other participants, whereas the leaderships of corporations and governments collude in ways that benefit themselves but harm less well-coordinated citizens, customers, employees and the general public all the time.

All three arguments are important and valid, but all three arguments lead to some interesting and different conclusions once you start thinking about protocol decisions with the three individual perspectives in mind. Let us try to expand out each of these arguments one by one.

Regarding fault tolerance, the core argument is simple. What's less likely to happen: one single computer failing, or five out of ten computers all failing at the same time? The principle is uncontroversial, and is used in real life in many situations, including jet engines, backup power generators particularly in places like hospitals, military infrastructure, financial portfolio diversification, and yes, computer networks.

However, this kind of decentralization, while still effective and highly important, often turns out to be far less of a panacea than a naive mathematical model would sometimes predict. The reason is common mode failure. Sure, four jet engines are less likely to fail than one jet engine, but what if all four engines were made in the same factory, and a fault was introduced in all four by the same rogue employee?

Do blockchains as they are today manage to protect against common mode failure? Not necessarily. Consider the following scenarios:

- All nodes in a blockchain run the same client software, and this client software turns out to have a bug.
- All nodes in a blockchain run the same client software, and the development team of this software turns out to be socially corrupted.
- The research team that is proposing protocol upgrades turns out to be socially corrupted.
- In a proof of work blockchain, 70% of miners are in the same country, and the government of this country decides to seize all mining farms for national security purposes.
- The majority of mining hardware is built by the same company, and this company gets bribed or coerced into implementing a backdoor that allows this hardware to be shut down at will.
- In a proof of stake blockchain, 70% of the coins at stake are held at one exchange.

A holistic view of fault tolerance decentralization would look at all of these aspects, and see how they can be minimized. Some natural conclusions that arise are fairly obvious:

- It is crucially important to have multiple competing implementations.
- The knowledge of the technical considerations behind protocol upgrades must be democratized, so that more people can feel comfortable participating in research discussions and criticizing protocol changes that are clearly bad.
- Core developers and researchers should be employed by multiple companies or organizations (or, alternatively, many of them can be volunteers).
- Mining algorithms should be designed in a way that minimizes the risk of centralization
- Ideally we use proof of stake to move away from hardware centralization risk entirely (though we should also be cautious of new risks that pop up due to proof of stake).

Note that the fault tolerance requirement in its naive form focuses on architectural decentralization, but once you start thinking about fault

tolerance of the community that governs the protocol's ongoing development, then political decentralization is important too.

Now, let's look at attack resistance. In some pure economic models, you sometimes get the result that decentralization does not even matter. If you create a protocol where the validators are guaranteed to lose $50 million if a 51% attack (ie. finality reversion) happens, then it doesn't really matter if the validators are controlled by one company or 100 companies — $50 million economic security margin is $50 million economic security margin. In fact, there are deep game-theoretic reasons why centralization may even *maximize* this notion of economic security (the transaction selection model of existing blockchains reflects this insight, as transaction inclusion into blocks through miners/block proposers is actually a very rapidly rotating dictatorship).

However, once you adopt a richer economic model, and particularly one that admits the possibility of coercion (or much milder things like targeted DoS attacks against nodes), decentralization becomes more important. If you threaten one person with death, suddenly $50 million will not matter to them as much anymore. But if the $50 million is spread between ten people, then you have to threaten ten times as many people, and do it all at the same time. In general, the modern world is in many cases characterized by an attack/defense asymmetry in favor of the attacker — a building that costs $10 million to build may cost less than $100,000 to destroy, but the attacker's leverage is often sublinear: if a building that costs $10 million to build costs $100,000 to destroy, a building that costs $1 million to build may realistically cost perhaps $30,000 to destroy. Smaller gives better ratios.

What does this reasoning lead to? First of all, it pushes strongly in favor of proof of stake over proof of work, as computer hardware is easy to detect, regulate, or attack, whereas coins can be much more easily hidden (proof of stake also has strong attack resistance for other reasons). Second, it is a point in favor of having widely distributed development teams, including geographic distribution. Third, it implies that both the economic model

and the fault-tolerance model need to be looked at when designing consensus protocols.

Finally, we can get to perhaps the most intricate argument of the three, collusion resistance. Collusion is difficult to define; perhaps the only truly valid way to put it is to simply say that collusion is "coordination that we don't like". There are many situations in real life where even though having perfect coordination between everyone would be ideal, one sub-group being able to coordinate *while the others cannot* is dangerous.

One simple example is antitrust law — deliberate regulatory barriers that get placed in order to make it more difficult for participants on one side of the marketplace to come together and act like a monopolist and get outsided profits at the expense of both the other side of the marketplace and general social welfare. Another example is rules against active coordination between candidates and super-PACs in the United States, though those have proven difficult to enforce in practice. A much smaller example is a rule in some chess tournaments preventing two players from playing many games against each other to try to raise one player's score. No matter where you look, attempts to prevent undesired coordination in sophisticated institutions are everywhere.

In the case of blockchain protocols, the mathematical and economic reasoning behind the safety of the consensus often relies crucially on the uncoordinated choice model, or the assumption that the game consists of many small actors that make decisions independently. If any one actor gets more than 1/3 of the mining power in a proof of work system, they can gain outsized profits by selfish-mining. However, can we really say that the uncoordinated choice model is realistic when 90% of the Bitcoin network's mining power is well-coordinated enough to show up together at the same conference?

Blockchain advocates also make the point that blockchains are more secure to build on because they can't just change their rules arbitrarily on a whim whenever they want to, but this case would be difficult to defend if the developers of the software and protocol were all working for one company, were part of one family and sat in one room. *The whole point* is that these systems should not act like self-interested unitary monopolies. Hence, you can certainly make a case that blockchains would be more secure if they were more *discoordinated*.

However, this presents a fundamental paradox. Many communities, including Ethereum's, are often praised for having a strong community spirit and being able to coordinate quickly on implementing, releasing and activating a hard fork to fix denial-of-service issues in the protocol within six days. But how can we foster and improve this good kind of coordination, but at the same time prevent "bad coordination" that consists of miners trying to screw everyone else over by repeatedly coordinating 51% attacks?

There are three ways to answer this:

- Don't bother mitigating undesired coordination; instead, try to build protocols that can resist it.
- Try to find a happy medium that allows enough coordination for a protocol to evolve and move forward, but not enough to enable attacks.
- Try to make a distinction between beneficial coordination and harmful coordination, and make the former easier and the latter harder.

The first approach makes up a large part of the Casper design philosophy. However, it by itself is insufficient, as relying on economics alone fails to deal with the other two categories of concerns about decentralization. The second is difficult to engineer explicitly, especially for the long term, but it does often happen accidentally. For example, the fact that bitcoin's core developers generally speak English but miners generally speak Chinese can be viewed as a happy accident, as it creates a kind of "bicameral" governance that makes coordination more difficult, with the side benefit of reducing the risk of common mode failure, as the English and Chinese communities will reason at least somewhat separately due to distance and communication difficulties and are therefore less likely to both make the same mistake.

The third is a social challenge more than anything else; solutions in this regard may include:

- Social interventions that try to increase participants' loyalty to the community around the blockchain as a whole and substitute or discourage the possibility of the players on one side of a market becoming directly loyal to each other.
- Promoting communication between different "sides of the market" in the same context, so as to reduce the possibility that either validators or developers or miners begin to see themselves as a "class" that must coordinate to defend their interests against other classes.
- Designing the protocol in such a way as to reduce the incentive for

validators/miners to engage in one-to-one “special relationships”, centralized relay networks and other similar super-protocol mechanisms.
- Clear norms about what the fundamental properties that the protocol is supposed to have, and what kinds of things should not be done, or at least should be done only under very extreme circumstances.

This third kind of decentralization, decentralization as undesired-coordination-avoidance, is thus perhaps the most difficult to achieve, and tradeoffs are unavoidable. Perhaps the best solution may be to rely heavily on the one group that is guaranteed to be fairly decentralized: the protocol’s users.

Like what you read? Give Vitalik Buterin a round of applause.

From a quick cheer to a standing ovation, clap to show how much you enjoyed this story.