UNFINISHED
DRAFT!

The DCS Theorem

Greg Slepak

October 4, 2017

Abstract. We present a probabilistic proof of the *DCS Triangle* [1][2]. We use the triangle to show decentralized consensus systems cannot cannot scale to support the transactional demands of centralized consensus systems, and therefore any *single system* can have *Decentralization*, *Consensus*, or *Scale*, but not all three properties simultaneously.

Definitions

A *system* is defined as any set of components (see Scope) following *precise rules* in order to provide service(s) to the users of the system. These services constitute the system's *intended behavior*.

In other words, a system S consists of a set of components, called its *scope* $\{S\}$, and a program ("state transition function", f_S), that together define the system's *intended behavior*, which means: upon receipt of message m, S uses f_S to update the internal state from s to s' and send back reply y within a time interval S_τ.

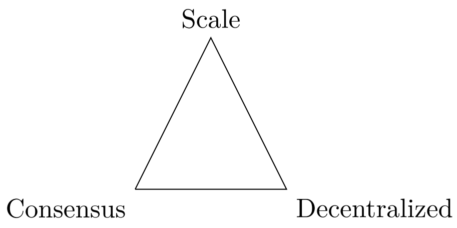
$$S(t) = \begin{cases} \{S\} & = \{component_1, component_2, \cdots\} \\ f_S(m,s) & = \{s', y\} \end{cases}$$

We note, additionally:

- The intended behavior of *centralized* systems is defined by its owners and can be modified without explicit consent from users of the system
- The intended behavior of *decentralized* systems cannot be modified without active approval from at least 50% of all *users* of the system
- The *scope* $\{S\}$ may change over time, but there are always several components of a consistent *type* (i.e. "all systems always have at least one *CPU*, one *developer*, and one *user*")
- The system's state s includes all data necessary for the system to compute f_S given a message m
- S is considered *compromised* if it fails to perform its intended behavior within the interval S_τ

- The system's users are always included in $\{S\}$ (a system without users is a system without messages or purpose)

We will proceed to prove that any single such system may possess, at most, two of three properties:



- **Consensus** means the system's state, s is a *shared state* that is updated by nodes running a *consensus algorithm* over a network, and that furthermore, the output of *consensus algorithm* determines the network's accepted output of f_S, and whether or not f_S completes within S_τ.
- **Scale** means the system is capable of updating its state at the same (or greater) rate as any competing system providing the same service to the same arbitrary set of users across the globe (*"at scale"*).[1]
- **Decentralized** means the system has no *single point of failure or control* (SPoF). Another way to state this is: the system continues to perform its intended behavior if any single element is removed from $\{S\}$, and no single component in $\{S\}$ has the power to redefine f_S on its own.

Systems whose intended behavior can be modified without the consent of their users are considered *centralized* due to the presence of a central point of control over the definition of the system.

Decentralization Scope & Relativity of Decentralization

Implicit in our definition of a *decentralized system* is the idea that the system is not compromised. A non-functioning system does not fulfill its intended behavior, and therefore, by our definition, is not decentralized.

Imagine a decentralized system S, whose intended behavior (its purpose) is to maintain the integrity of a database while being responsive to queries. It does so by attempting to eliminate all single points of failure within a given *scope.*

> The ***scope*** of a system refers to all subcomponents and all entities "reasonably relevant" to a system's functioning.

[1]Examples of "services" include: streaming video, sending messages, maintaining balances on a ledger, etc.

If we consider the scope of our "decentralized" database to be a computer with two CPUs and two hard disks (one primary, another backup), then we can say S is "decentralized" at $t = 0$ (has no single point of failure). However, if at $t = 1$ one of the hard disk fails, it is no longer decentralized since now there does exist a single component capable of compromising the entire system.

This means:

- Whether or not a system is decentralized can change over time.
- Any system can be called "decentralized" if we define the scope narrowly enough.
- All decentralized systems can be called "centralized" if we define their scope broadly enough.[2]

The narrowing and enlarging of the scope is called the *relativity of decentralization*, and it is why first choosing a reasonable and agreeable definition for a system's scope is vital before deciding whether or not it is "decentralized".

Computational throughput of consensus systems

The *computational throughput* $C(S)$ of any consensus system depends on three factors:

1. The computational power of each *consensus participant* (those able to select transactions which are *written* to the shared state).[3]
2. The amount of time the consensus algorithm considers messages to be lost (the *timeout* period).
3. The consensus *threshold* the consensus algorithm uses to decide whether consensus has been reached (e.g. "how big of a quorum is required").

[TBD. More info here about how the throughput changes depending on the computational power of members.]

Coordination costs

[TBD. Explain what coordination costs are.]

The difficulty for one entity to engage with another and work together toward a common goal.

Example: when Bitcoin was first launched, it would be difficult for any miner, to find other miners in order to create a cartel with 51%+ of the hashing power, simply because there were so many random miners all over the world.

[2]The entire Internet could be considered centralized if we include the entire solar system as part of the scope. The "single point of failure" could be the Earth itself, its atmosphere, the Sun, etc. Or, perhaps in the not distant future, a single ISP.

[3]Note that participants who are only allowed to *verify* state, but not participate in *creating* it, are not considered consensus *participants*. Instead, they are called state *verifiers*.

If there's, however, a small number of miners, then it is easier to (1) identify them, and (2) bring them together in a single room to coordinate around some goal.

Proof

We seek to prove the following restatement of the DCS Triangle:

> **Theorem.** Decentralized consensus systems that scale to meet the demands of competing (and functionally equivalent) centralized consensus systems, become centralized.

Our proof is built on the following axioms:

> **Axiom 1.** Access to computational power is not distributed uniformly, but is instead distributed according to rules resembling power laws. In other words, in any large population, a small group has access to computational power greatly exceeding what is available to the majority of the population.[4]
>
> **Axiom 2.** In centralized consensus systems, there exists an entity in $\{S\}$ with the power to exclude other entities from joining $\{S\}$ and participating in consensus, whereas in decentralized consensus systems no such entity exists.

[This is where the paper currently ends. What follows below are "brain dumps" of random thoughts about how to go about proving the theorem. I expect the entire paper to be no more than 5 pages long.]

OLD STUFF - Outdated brain dumps [To be deleted]

Our definitions do not allow us to write a definitive proof but they do allow for a probabilistic proof. Similarly, the choice of scope cannot be clearly defined for arbitrary systems, but must be arrived at by probability of what is "likely" to be a "reasonable" or "characteristic" scope of these systems.

So we start with a simple model of two systems of 1000 nodes in consensus with each other and of random compute power and observe:

1. High throughput (scale) implies few elite consensus nodes, no matter how they're arrived at.
2. Few consensus nodes implies low coordination costs.
3. Low coordination costs implies higher probability of failure (censorship).
4. High coordination costs + more consensus participants implies lower probability of failure (censorship) and also no scale.

[4]Consider how many of your friends have access to super computers.

The power to exclude slow nodes, must not exist in decentralized systems.

- The more users a system has the more valuable it is
- The more valuable a system is the higher the reward is for controlling it
- The lower coordination costs the easier it is to control a system
- The more valuable a system is, the more incentive there is for a controlling group to censor it, and the cost of not censoring it is high
- If a system has scale it is more valuable than a system that does not have scale.

If the coordination costs are sufficiently low enough (define, possibly with an assumption), and the $rewards - costs$ (profit) for colluding are high enough (define, possibly with an assumption), then the consensus group acts as a single entity with high probability, creating a single point of failure. At the very least, we can be *certain* that it is *much more likely* to act as a single entity than in a situation where coordination costs are high.

This is a "probabilistic analysis" (and not a very specific one) — not a proof.

Relationship between consensus and f_S

We have, up to this point, made it very clear that for f_S to remain uncompromised and fulfill its *intended behavior*, it must produce expected output in a reasonable time. If it fails, it ceases to fit our definition of being a single, unique and consistently identifiable system.

There is, in other words, the opportunity for users of the system to at one point see system A that processes their transactions, behaves as they expect, and moments later experience a *functionally different* system B that ignores their messages or otherwise behaves entirely differently.

The presence of a consensus algorithm in S introduces a dangerous hazard into our definitions and assumptions. Various results from distributed systems, like the FLP impossibility result and the CAP theorem, point out that distributed systems cannot always fulfill this demand. Messages can be dropped due to mysterious network outages, etc.

Under these circumstances it is not always possible for the system to maintain a single consistent shared state, even in highly centralized, controlled environments.

Furthermore, in a *decentralized* system that employs consensus, there cannot be a gatekeeper that decides who the consensus participants are, for such a gatekeeper would have the potential to remove all participants, or choose only participants who suddenly ignore the messages of existing users, and therefore they would represent a single point of failure.

The intuition behind our theorem is that in order for a consensus system to be decentralized, it must be more permissive as to who the consensus participants are, this permissiveness reduces the transaction capacity of the system as a whole (not everyone can run a data center at home).

The difference between a "*distributed* consensus system" and a "*decentralized* consensus system" is implied in the name. Decentralized consensus systems do not have a "center" deciding who the consensus participants are allowed to be.

Proof stuff

---

with n_t nodes at time t, all running f_S, and a second similar system S_2, except unlike S_1, the system employs a *consensus algorithm* to update its shared state.

Each system has At time t_0, running on n nodes, where the *average computational power* of nodes is c_{t_0}.

We define the *transactional capacity* of a system, $T_c(S)$, as the maximum transaction rate a system can sustain *at scale* without dropping messages.

Note the

Consider a universe U consisting of n entities, each of which we will label e_i.

Within U we define a system S that has a consensus process, meaning that the system's participants come to agreement about the system's state $\mathit{state}(S)$ at interval τ.

Each consensus participant bears a computational load c that is a function of the number of users that are sending messages through the system.

So the system S has:

- n_u number of users, each of which is sending some number of transactions per day. For simplicity's sake we will assume that each user sends an average of X transactions per day.
- n_c number of consensus participants, each bearing a load c_c that is a function of n_u, likely with some cap beyond which the system simply begins adding transactions to a backlog.

Each user has a computational capacity of c_u. The condition $c_u = c_c$ means that all users of the system, including consensus participants, have equal computational capacity. We note that each additional new user added to the system increases the number of transactions that the system has to process, and that there exists a number beyond which the transactional load exceeds c_u. At that point, any additional users added to the system will result in transactions being added to a backlog.

To clear the backlog, either the number of users has to go down, or the computational capability of each consensus participant has to go up.

Increasing the computational capacity of consensus participants while preserving $c_u = c_c$ implies a simultaneous forced hardware upgrade of all users of the system. At global scale, this is unheard of. Hardware upgrades do happen, but they are not simultaneously enforced globally. Therefore there now exists the condition that $c_c > c_u$, meaning that not all users can participate in the consensus process any longer.

If the system gains users at a rate that exceeds the rate at which users of the system can upgrade their computational capacity, then n_c/n_u will continue to become smaller and smaller.

We further note, that even if global computational capacity upgrades kept pace with the number of new users entering the system, each new *consensus participant* adds $t(n_c)$ *additional* time to the amount of time it takes consensus participants to reach consensus. We note that physics puts an upper bound on the amount of time it takes messages to propagate through the system, and therefore there exists an upper theoretically limit to the number of consensus participants, even assuming infinite computational capacity.

If $\sum t(n_c) > \tau$, the system will again have a backlog of messages.

Therefore we note two types of limits:

- A fundamental physical limit based on the speed of light that exists irrespective of computational power and is a function purely of n_c
- A practical limit that is a function of n_c, the rate at which computational upgrades occur in the system (R_c), and the rate at which users are added to the system (R_u).

We note that these limits only exist when there is a need for consensus. If there are no consensus participants (because there is no consensus), then transaction limit of the system is now "parallelized" instead of "serialized".

We note that if the ratio R_c/R_u is such that new users are joining the system faster than they are able to upgrade their hardware to meet the increasing computational demands of consensus, then n_c, the number of consensus participants, becomes increasingly smaller with respect to n_u.

Here is a graph using Moore's law:

If we attempt to scale the system quickly, the triangle says that we will either lose Consensus or Decentralization. Why?

Well, if the system remains decentralized …

If the system has consensus then …

We see that as time goes on, the system is no longer able represent each user. In effect they are not "counted", and therefore D_2 is never met. If only 1% of users can participate in consensus, then there is no way to "count" their votes. Whether the system uses Proof-of-work or Proof-of-stake is irrelevant, because consensus participants determine which transactions are allowed to be "heard" (included) in the consensus-process, and they are free to simply ignore any "votes" of non-consensus participants.

Characteristics of decentralized systems

Note that our definition of *decentralized* necessarily implies various characteristics about the system.

Permissionless

Permission to run f_S implies the existence of a gatekeeper (or gatekeepers). In practice this can manifest as closed source software, licenses, DRM, etc.

In our terminology, "at scale" means the system must support a global and *arbitrary* set of users. This means, if the system operates at scale,

Appendix A: The "DSS" Triangle

Scale-Security-Decentralization. Just another name for the same thing, but poorly defined.

Appendix B: Examples of all three types of systems

Decentralized consensus: Centralized consensus:

Appendix C: Errata from "Slepak's Triangle"

- Note name change
- Note I meant subset not superset

Characteristics of decentralized systems

To understand the proof, we must understand the characteristics of decentralized systems.

Low-cost of participation

Decentralized systems typically have a low-cost of participation. In other words, little effort is needed to use the system, and anyone can play any role.

High costs usually point to the existence of a privileged entity with the power to exclude others from participation (a form of censorship). Such an entity represents a single point of failure (D_1) that could prevent the system from fulfilling its intended purpose for most of its users.

Permissionless and inclusive

Our definition for decentralization means there is no trusted third-party deciding who can or cannot participate. Anyone around the world can join the system as long as they meet very basic resource requirements (e.g. an Internet connection).

Gatekeepers represent a central point of control, a violation of D_1 and D_2.

Most importantly, decentralized systems do not exclude inefficient participants. Rather, they go out of their way to ensure the most amount of participation. This is what keeps decentralized systems decentralized, as otherwise economies of scale will push the cost of participation up until a controlling group emerges, creating a single point of failure (D_1), along with the ability for that group to dictate behavior to the rest of the system (D_2).

This does not imply that decentralized systems are slow, but it does mean that decentralized **consensus** systems are *always* significantly slower than their centralized counterparts.

Censorship-resistant

The permissionless nature of decentralized systems, and their lack of a central point of control or failure, means they inherently resist all attempts at censorship.

Can centralize over time

Protocols and *implementations of protocols* are two different things. A protocol can only provide *the ability* for a decentralized system to exist, it does not provide a guarantee.

All decentralized systems, even those without consensus, can be centralized if steps are not taken to combat their centralization. Single points of failure are likely to emerge as the system gains more users and interacts with the systems around it.

Decentralized systems *involving consensus* are especially vulnerable to centralization. An increase in the number of users represents two fundamental obstacles to their decentralization:

1. The system's shared resource becomes more valuable as it gains users, which increases the reward for successfully compromising the system. Furthermore, the system acts as a threat to the value managed (or monopolized) by established centralized players. Combined, these factors incentivize attackers to either find or create a single point of failure in the new system.
2. More users means more diversity of opinion over the system's future direction and the fate of its shared resource. Simultaneously, it becomes more difficult to distinguish real users from fake sybils or deliberate attempts at sabotage. As the distance between the system's maintainers and its average user increases, so too do misunderstandings. It becomes increasingly likely that *any* decision over the fate of the system will result in the alienation of a significant fraction of users.

Examples

A Narrow Proof

We will construct a proof based on our very narrow definition of the term **"mainstream"**, which—it must be emphasized—we feel is *reasonably close but not identical to* various colloquial understandings of the term.

It is important to understand what we *are not* asserting:

- We **do not assert** that it is impossible for a decentralized consensus system to reach mainstream adoption. It is certainly possible:
- One could create a second system along the *decentralized-mainstream* edge of the triangle and tie it to the first. This is what Bitcoin's Lightning network does.
- It might be possible to connect multiple decentralized-consensus systems to each other while maintaining the decentralization of the system-of-systems, although this is still unproven. This is what the Mauve paper[2] and Tree-Chains[3] attempt to do.
- More extreme measures include reducing the world's population to the point where "Mainstream" means "10 people" and therefore the difference between decentralized consensus systems and their centralized counterparts is negligible.
- We **do not assert** that decentralized consensus systems of *the future* cannot compete with the centralized systems of *the present.* Certainly, as technology improves around the world the performance all systems improves. However, if one were to bring technology from the future to the present, that technology would improve the performance of centralized systems just the same.

The proof

We will construct our proof by making three assertions:

1. *Decentralized consensus systems* cannot process information as quickly as their centralized counterparts.
2. *Decentralized consensus systems* have an upper user limit, which, if exceeded, makes the system increasingly centralized.
3. If the previous two assertions are true, the triangle holds.

1. Decentralized consensus systems are slower

TODO: read: - https://jvns.ca/blog/2016/11/19/a-critique-of-the-cap-theorem/ - https://www.cs.cornell.edu/courses/cs614/2006fa/Slides/FLP_and_Paxos.pdf - http://book.mixu.net/distsys/abstractions.html

A consensus system S, whether it is decentralized or not, is comprised of consensus participants p (nodes). As before, the time it takes for those nodes to come to consensus on the system's state is its *period*, τ.

The difference between a centralized consensus system $\mathbb{C}$, and a decentralized consensus system $\mathbb{D}$, is that membership in $\mathbb{D}$ is not pre-determined. In other words, the members $p \in \mathbb{D}$ are not specified by any single entity (that is what it means for the system to be *permissionless* and *inclusive*). Otherwise, the system would violate D_1 and D_2.

At the same time, the inclusive nature of the decentralized consensus system cannot mean that any single participant is capable of preventing consensus. If that were the case, the system would again violate D_1 and D_2.

We observe that in both $\mathbb{C}$ and $\mathbb{D}$, each participant p takes *validation time* V_p to process and validate the messages it receives (determined by its computation capability), and adds an additional coordination cost/overhead O_p to τ, the amount of time it takes the system to reach consensus (determined by the connections between participants).

Next, we say that the system reaches consensus once some threshold (greater than 50%) of participants agree on the system's state.

We define M as the set of messages the system can process during τ.

Clearly, given two systems, S_1 and S_2, identical in all respects except by the speed with which members process and relay messages, the system whose members are slower will process fewer messages.

Now, let us consider two hypothetical "fastest possible" systems, $\mathbb{C}$ (centralized) and $\mathbb{D}$ (decentralized).

Let us consider $\mathbb{C}$ first. We note that if the system did not employ consensus (e.g. $N = 1$), then the speed at which the system could process messages would be fully determined by V_{p_0}, and therefore p_0 would have to represent the "fastest available hardware" for a single node.

Extending this to $\mathbb{C}$, where $N \geq 2$, we observe that for any given M, the "fastest possible" centralized consensus system would have to consist of some optimal number of nodes, N, beyond which the coordination costs incurred by adding an extra node reduce the system's performance as a whole.

Therefore a lower-bound for τ is defined as:

$$\tau \geq \sum_{i=0}^{N}(p_i)$$

Given the same high transactional demands (e.g. 1 million messages), we denote $\mathbb{C}_\tau$ as the time taken by the fastest possible centralized consensus system to process those messages, and $\mathbb{D}_\tau$ as the time taken by the fastest possible decentralized system.

How do we know when the system has reached consensus to begin with?

We can start with the trivial case of there being a *leader* node, p_1, that determines the value that the system is coming to agreement on. Obviously, if p_1 is faulty, the system will never reach consensus because all the other nodes depend up on it.

Since each node in the system

As mentioned, decentralized systems must be *permissionless* and *inclusive*, otherwise they violate D_1 and D_2.

This coordination cost increases as more nodes are added as participants in the consensus.

If consensus is added on top of a decentralized system, the requirement for inclusivity does not go away. Thus the system must allow even "inefficient" participants to participate.

A *centrally managed* distributed system can ensure that all consensus participants meet a certain minimum processing capabilities by virtue of there being a single point of control over the system (a single point of failure). Decentralized consensus systems do not have this single point of failure, but in giving it up they also give up the efficiency gains

As suggested by Liebig's law of the minimum[5], and as demonstrated by Croman *et. al.*[3] in their analysis of blockchain scaling, the growth (or size) of a decentralized consensus system is limited by the scarcest resource.

[5]https://en.wikipedia.org/wiki/Liebig%27s_law_of_the_minimum

2. Decentralized consensus systems have an upper user limit

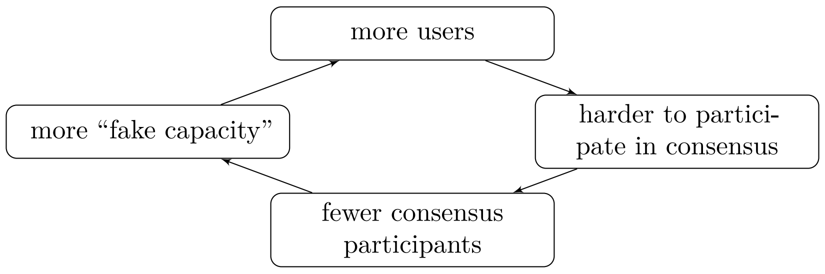
If the system enforces high resource requirements on participants in the pursuit of higher throughput, then the system become becomes increasingly centralized and the likelihood of central points of failure increases.

More users are forced to trust an increasingly smaller group to determine what the state of the system is. A deadly mixture emerges:

1. Reality starts to diverge from marketing claims the so-called "decentralized" system.
2. Simultaneously, the system's advocates point to "cheaper fees" due to "increased capacity".
3. News of "progress" and "cheaper fees" in this "decentralized" system attracts more users.

In reality, while *using* the system may be less expensive, *participating* in the system as an equally privileged node becomes significantly more expensive. The "increase" in capacity is also illusory, for by now the system has long exceeded its tolerable capacity for maintaining its decentralization.[6]

A feedback "death spiral" occurs:


more users
harder to participate in consensus
fewer consensus participants
more "fake capacity"

The Croman work expanded on prior research [@??] and showed that global bandwidth is, at present, the scarcest resource limiting Bitcoin's growth. Further, it shows that 4MB is the absolute.

3. Given (1) and (2), the triangle holds

Appendix A: Choosing a safe redefinition threshold

Link to flocking study and point out how 95% creates a central point of failure.

The more value being considered / the more potential harm, the higher the threshold must be.

[6]Threat of DDoS also goes up.

Because not all users can vote, because forks are high-stakes irreversible decisions[4], and because a split vote indicates strong community indecision/disagreement, it is usually better to use a higher threshold.

The reason for very high thresholds is due to the fact that most users of the system will be unable to vote because there's no way to safely "count" their votes.

Now we instantly see a relationship between Consensus and D_2. We also observe that "users" is ambiguous. In Bitcoin, is it token holders? Is it miners? Developers? How do you measure these?

However, we have a serious problem: we can't measure whether we've reached the threshold beyond ~150 users! We only assume we can. OTOH, with certain hierarchical counting methods, where there is both transparency and everyone is incentivized to verify that the vote is accurate, maybe you can. E.g. CNN had a centralized website showing the numbers for various precincts reporting in, and each presinct can check that their number is accurately represented. With CONIKS and/or PoW, it maybe be possible to use such a method to count more people.

One way to get around this limit is to reduce the amount of information-flux/change that the system is expected to handle. We can do this by making the system immutable (or mostly immutable). For example, religions are systems too, and they are capable of supporting so many "users" precisely because they are based on texts that are widely distributed and not only do not change, but are explicitly expected *not* to change.

It's 75% of *users* not stake or CPU!

PoS voting doesn't count if stake is centralized, and by Zipfs law we can expect it to be. Same for PoW.

A human being physically cannot verify a vote count involving more people than what they can visually distinguish in their visual field of view without running into the Sybil attack or PoS (e.g. $1-1-vote).

References

[1] T. McConaghy, "The DCS Triangle," 10-Jul-2016. [Online]. Available: https://medium.com/the-bigchaindb-blog/the-dcs-triangle-5ce0e9e0f1dc.

[2] G. Slepak, "Slepak's Triangle," *Rebooting Web-of-Trust*, 17-Oct-2016. [Online]. Available: https://github.com/WebOfTrustInfo/rebooting-the-web-of-trust-fall2016/blob/master/topics-and-advance-readings/Slepaks-Triangle.pdf.

[3] K. Croman *et al.*, "On Scaling Decentralized Blockchains," 2015.

[4] G. Slepak, "Deprecating May's Theorem," *Group Income*, 2016. [Online].

Available: https://groupincome.org/2016/09/deprecating-mays-theorem/.