Data & Policy (2025), 7: e10
doi:10.1017/dap.2024.50

COMMENTARY

Majoritarianism and Monoculture (M&M)

Mansoor "Manny" Ahmed[1], Dann R. Toliver[2] and Hazem Danny Nakib[2]

[1]OpenOrigins Limited and Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom
[2]Centre for Redecentralisation (CRDC), Department of Computer Science and Technology, University of Cambridge, Cambridge, United Kingdom
Corresponding author: Mansoor "Manny" Ahmed; Email: mansoor@openorigins.com

Received: 02 June 2022; **Revised:** 28 April 2024; **Accepted:** 15 July 2024

Keywords: consensus algorithms; digital collaboration; decentralisation; majoritarianism; minority rights

Abstract
Research in decentralized computing, specifically in consensus algorithms, has focused on providing resistance to an adversary with a minority stake. This has resulted in systems that are majoritarian in the extreme, ignoring valuable lessons learned in law and politics over centuries. In this article, we first detail this phenomenon of majoritarianism and point out how minority protections in the nondigital world have been implemented. We motivate adding minority protections to collaborative systems with examples. We also show how current software deployment models exacerbate majoritarianism, highlighting the problem of monoculture in client software in particular. We conclude by giving some suggestions on how to make decentralized computing less hostile to those in the minority.

Policy Significance Statement
Decentralization is an age-old question of importance to policymakers. Traditionally, it had to do with structural changes to government agencies and the degree of decision-making that is concentrated in certain entities or branches of government; whether it is more spread out or pushed closer to those that the decision impacts. The recent trend of decentralized computing provides an opportunity to decentralize existing government services and public infrastructure in a way that has not been possible before. As governments become digital, what has been called "e-government," the decentralisation question is resurfacing. Our article highlights overlooked questions regarding the drawbacks of introducing decentralized computing into government by leading to antithetical outcomes, further centralization and majoritarianism that can be detrimental to the interests and rights of minorities, and possible solutions for policymakers to consider.

1. The problem
There is a form of governance in the world today that affects many people and has many supporters. It is democratic in principle, in that each decision requires a vote, but those who do not side with the majority are ignored, and their protests are erased from history. Nothing they do can change this, and continuing to protest may result in being exiled for doing the work of the Adversary.[1] The underlying assumption is that anyone that does not agree with the majority is incorrect, dishonest, and compromised:

[1] And struck from history: a kind of damnatio memoriae.

in other words, an agent of evil. As dystopian as this may seem, it is exactly the way consensus algorithms work today.

This harmful majoritarianism is built into consensus algorithms by design. The term for a dissenting minority in the literature is "adversary," and the goal of consensus algorithms is to withstand such an adversary. This *adversarial assumption* is explicit within every consensus algorithm. Rejecting and silencing dissenting voices limits the situations where these systems can be used and exacerbates other risks.

A major risk that suppressing dissenting voices introduces in computational systems is that of *software monoculture.* Decentralized systems tend to have multiple client versions that any potential user of the network can choose from. This is often done not just as a means to expand compatibility but also as a means to avoid a single point of failure. Unfortunately, it is often the case that a particular client is used by the vast majority of the network. Compromising the Go Ethereum client ends up being equivalent to compromising the Ethereum network. This example of software monoculture, which we return to later, shows one of the ways in which decentralized systems are susceptible to majoritarian takeover. It also helps highlight the challenge that applications running on these systems face. A smart contract on a blockchain has no exposure to dissenting voices that arose during the consensus process. It can only see the outcome of consensus, which appears to be a smooth featureless sequence of blocks, while in reality there are many "orphaned" blocks and other forms of dissent. If the Go Ethereum client is compromised, all the honest clients could protest, but their voices go entirely unheard by the denizens of that system. If applications on blockchains could hear that dissent, they could make different decisions—for instance, to stop processing until the system stabilized. Without exposure to dissenting opinions, such measures are impossible.

This is in stark contrast to many real-world governance structures where minorities and minority opinions are afforded protections, and where their dissent is remembered and not just evaporated into the ether.[2] Recently, there have been proposals to move some governance operations onto smart contracts. This is concerning since even if we design these smart contracts to afford equivalent protections, they are built on a substrate of absolute majoritarianism (i.e. that of the consensus layer). If we are to move to a world of digital decentralized governance, then that world must be at least as good in the protections it affords to minorities, if not better.

In this article, we highlight our concerns about majoritarianism in computerized governance systems, contrast such systems with existing minority protections and suggest possible ways in which we can make these systems more resilient to majoritarian takeover.

2. Minority protections in real life

There are a variety of areas in governance, administration, conventions, standards, and law in different societies where the purpose or aim of the regulation is to carefully and specifically address a power imbalance between respective parties trying to accomplish something. Often this has to do with the relationship between, for example, a state and its citizen, or a corporation and their shareholders, or an individual user or consumer with any corporate seller. The rationale behind this is that often the set of circumstances between the parties disadvantages one party, whether because they have no other alternative or choice or they have fewer resources. Alternatively, it is thought to be a good thing for a particular right to be protected, for example, because the rule of law demands it (such as being treated equally before the law or being afforded access to legal representation).

There are a number of ways and areas in which a power imbalance may be addressed, primarily by way of the law in a particular jurisdiction, either through legislation, case law, or through standards bodies or agencies, and even industry codes of conduct. For the purposes of this article, we have selected four areas focused on legal solutions to addressing power imbalances. They are relevant because majoritarianism speaks particularly to a group (the majority) being overly powerful over a

[2] Or the doge.

minority that is, in many ways, helpless and voiceless, absent certain powerful protections that prevent them, and their rights from being dismissed, and potentially oppressed. This is valuable because, although code is not law, the thinking behind it can be leveraged on their own merits, and existing protections can cross over into the systems of discussion in this paper, encoded in the design of a network, or carried over and enforced into it.

2.1. Company law protections

The first area is in company law with regard to minority shareholder rights. For example, in England and Wales under the Company Act, all shareholders are granted certain basic rights such as the right to vote (UK Companies Act, 2006). Typically, the rights afforded to shareholders are modified by the shareholder agreement or the articles of association of the corporate entity, which may deviate from that which protects certain rights, or where rights or protections are vague.

Other rights provided by the Companies Act in England and Wales include, where there is a shareholder of 5% or more, the ability to circulate a resolution, require the company to call a general meeting, or prevent the reappointment of some auditor. Regarding 10% shareholding, the ability to mandate an audit by the company and call for a vote in a general meeting. For those shareholders with 25% or more of shareholding, the power to block a special resolution. Each of these enables a minority shareholder to play a greater role in the governance of the business than they would otherwise by providing them with powers to seek redress when their interests may be otherwise curtailed.

In the U.S., company law changes from State to State, though the majority of States include certain rights for minority shareholders. These include a fiduciary duty that majority shareholders owe minority shareholders to deal with minority shareholders in good faith. Other rights afforded to all shareholders include voting, reviewing the company's financial information, and attending certain meetings. Access to financial information is particularly relevant here, which a minority shareholder may demand and provide the reason for said demand by reason of transparency to understand the nature of their commercial activities, notwithstanding their minority shareholding.

2.2. Contract law protections

A second area where such protections find themselves is more generally in contract law, both in England and Wales and in the EU as it relates to unfair terms included in contracts. This has to do with contract terms that are deemed to be unfair between parties and found in the Unfair Contracts Terms Act 1977 as well as the Consumer Rights Act 2015 as it relates to contracts between consumers and organizations (UK Unfair Contract Terms Act, 1977) (UK Consumer Rights Act, 2015). These pieces of legislation highlight important principles of reasonableness and fairness, as well as types of terms in contracts that would be deemed unfair or unreasonable. For example, in Germany, consumer contracts cannot automatically have mediation clauses in order to protect consumers from unfair and possibly illegal practices that larger corporations are attempting to keep quiet and prevent being coupled together in court processes (and it is thought that mediation would provide consumers greater difficulty in seeking redress).

2.3. Human rights protections

The third way has to do with civil and social rights that are protected by law. There is no disputing that certain groups within society are harmed to greater and lesser extents than one another when it comes to discrimination, housing, welfare benefits, and freedom of speech and expression. Nonetheless, a set of protections, which encompass civil and social rights are afforded protection, whether by way of the Constitution of the U.S. and its Amendments, or by way of common law principles in England and Wales, coupled with the Human Rights Act 1998, which brings into English Law the European Convention on Human Rights set forth by the European Council across Europe. (UK Human Rights Act, 1998)

2.4. Civic participation

The fourth way has to do with the actual voting of the state in question. Democracy has long been heralded as a prerequisite for a fair and just society. Broadly speaking, a democracy in practice entails a majority vote for a particular political party thereby being the governing political party based on a written or unwritten constitution. However, in practice, there are a variety of iterations, such as a representative democracy in the U.S., or a more direct version such as those present in the Cantons of Switzerland. These are typically coupled with a variety of checks and balances and reviews, including in the passing of legislation through Congress and the Senate, or through the House of Lords and House of Commons in the U.K., where the House of Lords provides a second review stage of any piece of legislation. Notwithstanding this, there are a number of risks present for majoritarianism that is checked by robust legal protections to prevent a majority from curtailing a minority of the populations' rights and interests.

As we return to considering majoritarianism as it relates to different decentralized networks, what has become apparent in this section is threefold. First, there exist a number of existing regimes of law that have addressed questions of majority rule. At the same time, those have been in different circumstances, such as in commercial arrangements or within corporate entities. Second, there are a number of basic and fundamental rights and interests that are afforded protection no matter what. Third, there are layers of governance and administration aimed at ensuring the lack of abuse of power because, at the end of the day, that is the primary locus of any given mechanism to prevent majoritarianism. The common thread throughout each of these is identifying the ways by which to limit the abuse of power, something that is more acute in the contexts of clear imbalances of power.

3. What can go wrong?

The most obvious place where this consensus-level majoritarianism can have an adverse effect on real people is where these systems are being used for governance of real-world goods. An example of this are smart contracts used to track property ownership.[3] Current systems require a large degree of manual oversight in the form of sign-offs by multiple human-controlled accounts at the smart contract layer as well as human interaction off the chain for processing. However, should this be removed (as is required to enable a true DAO model) without protections against majoritarian takeover of both the smart contract and consensus layer then, we can expect this vulnerability to be exploited.

To make this example more concrete, let us consider what would happen if we deployed such a property tracking system without any off-chain human interactions. In this case, we can imagine that the property ownership is represented by a smart contract running on some blockchain, say, Ethereum. Presumably, the transfer of ownership would be represented by a transaction sent to that smart contract. Now, even if the smart contract were written perfectly, this system is at the mercy of third parties. An adversary could decide to censor a particular transaction from occurring by, for example, bribing a majority of mining pools. Without some fall-back off-chain mechanism, you would be incapable of selling your house irrespective of how many interested buyers you get.

3.1. Historical revision

A broader concern with majoritarianism in blockchain systems is the ability of an adversary to rewrite history. This attack vector has been acknowledged from the start in the form of so-called 51% attacks. In the rest of the world, we have channels that are at least somewhat outside of majoritarian control: a megaphone in the public square, printed flyers, skywriting, even viral messages on WhatsApp. Each of these actions can be taken immediately by an individual, resulting in the creation of an artifact whose forced disappearance is not entirely straightforward.

In the blockchain realm, we cannot rely on such channels when the majority can literally rewrite every aspect of shared history with no lasting artifacts. There is no common *impervious substrate.* And due to

[3] As piloted by HMRC: https://hmlandregistry.blog.gov.uk/2019/05/24/could-blockchain-be-the-future-of-the-property-market/

the monoculture of clients, a 51% style attack really only requires a compromise of one piece of software: not as daunting a task as consensus whitepapers may have one belief.[4]

These concerns apply to an even greater extent to assets that exist only inside these computational systems. In those cases, the nature of the asset itself, not just its record of ownership, is a manifestation of the collective, living memory and shared history of the computational system. The majoritarian faults in that memory that cause events to be ignored or forgotten do not merely change the information about the asset or add more information to it—they actually reach back through time and rewrite its history. From the perspective of a smart contract, there is no memory of the previous version of the asset. It is as though it never existed.

3.2. Application layer and ecosystem concerns

Majoritarian concerns extend to applications at the smart contract layer as well. Even if we assume that the consensus layer is resilient to takeover, if DAOs are not structured appropriately, they can be taken over with no remedial action possible. For example, the DigixDAO in 2020 was dissolved following a majority vote (Writer, 2020). The sizeable minority that dissented (or were simply offline for a period of time) were, for all intents and purposes, ignored with no compensation offered. In this way, the structure of current DAOs resembles a particularly harsh form of a "drag-along right" clause triggered at a 51% vote.

These decentralized systems do not exist in a vacuum. Different participants take up different roles to contribute to the system, participating in growing it, creating value, and building its rule-set, including its core consensus mechanisms. These systems touch against other systems, whether political, economic, social, and legal, each of which is comprised of its own rule sets and which sometimes are complimentary, and at other times conflict.

There may, for example, exist certain minority protections outside of the blockchain system, which are not protected within it. While such protections would be applicable if the laws of that relevant country applied, enforcing them in a decentralized network may prove particularly difficult. In some cases, such enforcement may in fact be near impossible due to the limited ability to force a decentralized system into adhering to, say, a court order, or the inability to identify the correct parties.

Such post-hoc enforcement mechanisms are generally worse than automatically building protections into the design of a system. Finding ways to carry existing protections into the blockchain system is of paramount importance in the design of the system. It is also possible that we may find additional solutions to the problems of majoritarianism and new forms of protection that are not present in traditional systems, opening opportunities to go beyond the level of the resilience of existing systems.

4. Possible mitigations

The problems of adversarial assumption are inherent to the deepest level of our current thinking around consensus algorithms. Computer systems are incredibly precise, and therefore, inherently fragile. Building a resilient system on such a substrate is rather difficult. Consensus systems therefore optimize for the following goals:

- Simplicity. Complexity generally makes things more fragile.
- Preservation. The essential invariants of the system cannot be broken.
- Progress. The system must continue making progress even under adverse conditions.

[4] An often proposed solution to attacks like this is to simply fork the blockchain. Sometimes forking is actually encouraged. Speculation is an inherently adversarial all-versus-all activity, and speculative assets can benefit from adversarial actions (hardforks tend to benefit token holders, for instance). No such benefits accrue for external assets that are merely being managed or represented in a computational system: no amount of hardforking will duplicate a house, or create more land.

This leads directly to the effects we detail: the simplest way to make progress while preserving the essential elements is to jettison everything else. The simplest way to continue in the face of unknown adversarial forces is to consider all disagreement to be adversarially motivated.

To mitigate the problems of majoritarianism, we must relax these design goals. We will categorize potential solutions by the primary goal they relax, though these relationships are not exclusive.

4.1. Progress

The first goal we consider relaxing is **progress**. Right now, the majority in a consensus system can unilaterally override even a very sizeable minority, completely obliterating their objections. In fact, this behavior is an explicit design goal because being able to silence larger groups increases the ability of the system to make progress in the face of a powerful adversary that has compromised many nodes.

This is predicated on the assumption that an adversary will have to pay a fixed cost to individually compromise each node. In real, deployed distributed systems, this is not usually the case due to the effects of software monoculture. Typically a single client (e.g., a Python client or Golang client) will dominate the distribution of clients across nodes.[5] An adversary that gains control of the codebase of that client can compromise a majority of nodes in a single action.[6]

In such a monoculture, an adversary does not need to pay a cost that scales linearly with the number of clients they compromise. It may be worth slowing things down in such cases, in order to allow a sizeable minority to veto or delay a decision by the majority.[7]

Letting a minority of nodes veto or delay a decision reduces the system's ability to make progress, but provides an important layer of protection. This protection extends beyond the problems of software monoculture to other cases where the majority is compromised, and even cases where the majority may have incentives that are misaligned with the goals of the system itself (including temporary incentives, where an adversary is for instance attempting widespread bribery).

4.2. Preservation

The second is by relaxing **preservation** of invariants. For example, one of the invariants observed by most distributed systems is that they do not discriminate on client type.

Weakening this is another way of addressing the risks of software monoculture, by imposing limitations on the proportion of nodes that can run a particular client. This can increase the client diversity in the system and reduce the risk of developing a software monoculture.

This comes at the cost of some extra gatekeeping to ensure the client software that a node is running is within the system's prescribed limits. While most nodes would probably honestly report which client they are using, there may be incentives for lying, perhaps even within the software clients themselves. One could try to make such dishonesty more cumbersome: requiring nodes to use trusted execution environments, for instance. This reduces the risk of lying, but does not eliminate it, given the range of demonstrated attacks against current TEEs (van Schaik et al., 2020; Götzfried et al., 2017; Brasser et al., 2017; Nilsson et al., 2020).

There are majoritarian threats that go beyond software monoculture, and other invariants we can weaken to address those. Many consensus systems not only classify dissenting opinions as adversarial threats but also eradicate those opinions from history as a kind of *damnatio memoriae.* This stems from requiring a single chain of updates, with no room for additional material to be added and no way to bring forks back together. Relaxing this constraint, for instance, by building upon confluent data structures, can leave the dissenting opinions available for analysis within the system.[8]

[5] As a representative example, at the time of writing, one client (geth) accounts for 83.7% of execution nodes[6].

[6] Note that codebases typically have deep dependency trees, and a compromise or compromisable bug in any dependency in that software supply chain could compromise the entire client.

[7] The irony that the adversarial assumption, whose whole reason for being is to reduce the risk of adversarial takeover, is itself a major risk factor for adversarial takeover is not entirely lost on the authors.

[8] By smart contracts, for instance, whose input is generally limited to the main chain within a blockchain system.

Weakening the invariants of a consensus system is no easy task, but doing it provides ways to address these threats by increasing diversity and transparency. When paired with the ability to block or delay decisions through relaxing progress, this can result in systems that are more resilient to threats in a rapidly changing environment and better suited to handling use cases beyond speculative assets.

4.3. Simplicity

Finally, we can consider mitigations that primarily relax **simplicity**. Current consensus systems are islands unto themselves.[9] This is in part explained by their use cases, which generally involve securing high-value assets, whether looking at a public blockchain or a Zookeeper instance in the backend of a distributed system. Those assets cannot be moved to another system (a bitcoin only exists inside Bitcoin), and the system operators bear the full burden of securing those assets in their system, so they must build thick walls. Any kind of integration with an external system will typically increase risks by punching holes in those walls.

Despite this, it is worth considering what could be done to mitigate majoritarian issues by having multiple systems work together. This is, in fact, the primary basis of protections in our nondigital lives. By drawing together multiple independent but interconnected systems, we achieve checks and balances that would be impossible in a single system, however well crafted.

Such ecosystems tie together incentives for their participants and reinforce rule-following behavior in a variety of ways. The degree to which they manage those two activities is probably a good indicator of their long-term success.

Designing computer systems that can achieve the same feat of making decisions, delegating responsibility, and managing resources through a complex web of interconnected and interdependent systems is hard to imagine. Yet we have examples all around us, from biological ecosystems to corporate governance to our civic lives.

There are several open questions that come up as we start looking at relaxing the levers above: What would it look like to have multiple consensus systems that each independently managed their own domain, but where those domains have significant overlap? What does it look like if systems are not always nestled hierarchically, like Zookeeper inside a data centre or a DAO inside a blockchain, but one system may be part of several other systems, as well as having some portion of independence? How do our systems of today change if the assets they manage can be selectively moved by their owners to another system with respect to state management? Questions like these may lead to distributed system designs that are more resilient in the real world.

5. Conclusion

We have outlined the problem of majoritarianism, an issue that has been under-researched and remains unaddressed. Majoritarianism is built into consensus algorithms. Unlike existing political and legal systems, consensus systems maintain no protections for parties with dissenting opinions, and provide no channel for the minority to voice themselves other than by leaving or forking the system. This is exacerbated by the tendency of decentralized systems toward a monoculture in code and in choice of client, which in turn increases the risk of an adversary being able to take over the system.

We provided an overview of ways this problem is addressed in real life, along with examples of how it impacts technical systems. In particular, we highlighted software monoculture as a large source of risk and described how the design goals for consensus systems are predicated on an unrealistic assumption that each node will be equally hard to compromise. This leads directly to attempting to maximize the number of nodes that can be compromised before system failure, and to the adversarial assumption: that all dissenting nodes are adversarial and must be ignored.

[9] Readers unaccustomed to seafaring may substitute “walled gardens” as their metaphor of choice.

This focus on optimizing exclusively for the ability of the system to make progress under maximum adversarial load while preserving its invariants leads to a focus on simplicity, preservation, and progress. By relaxing these, we can create systems that are resilient to the systemic threats of software monocultures and that can make progress in the face of incentive misalignments by providing those in the minority with options beyond quitting or forking the system.

We divided these mitigation techniques into three broad classes based on which goal they primarily relaxed. By relaxing progress, the system becomes slower when there is a large amount of dissent, but of course that is exactly when you want the system to slow down, so those involved can determine the source of that dissent and how to respond to it, particularly in the case of a bug or compromise in a popular client or library.

We considered several invariants whose preservation could be relaxed and saw how this could bring an increase in diversity and transparency to help address the issues of monoculture and majoritarianism. Finally, we considered what software systems in the future might look like as we find the techniques needed to create resilient systems that can interconnect and interdepend in complex ecosystems, creating richer and more adaptive solutions than a rigid adherence to simplicity would allow.

Each of these techniques provides an opportunity to rethink the assumptions we make as system designers and the properties on which the resulting systems rely, giving us a chance to directly address the issues of monoculture and majoritarianism.

Majoritarianism in consensus systems must be addressed in order for consensus algorithms to handle the use cases that have been proposed for them, including those in the public sector as well as mainstream private sector uses. The rights presently enjoyed and protected by people are at risk of being contravened through the introduction of technology built on a shallow and draconian adversarial assumption. To build technological systems that can help provide good governance, we must incorporate good governance into the core of those technological systems.

Funding, interests, and data availability disclosure. This work received no specific grant from any funding agency, commercial, or not-for-profit sectors. The first and second authors are involved in companies building on top of and developing decentraliszed technologies (OpenOrigins and TODAQ, respectively). There is no data associated with this paper.

Author contribution. Mansoor "Manny" Ahmed: Conceptualization, Writing - Original draft preparation. **Dann Toliver**: Conceptualization, Writing - Original draft preparation. **Hazem Danny Nakib**: Writing - Original draft preparation, Reviewing and Editing.

References

Brasser F., **Müller U.**, **Dmitrienko A**, **Kostiainen K**, **Capkun S and Sadeghi A-R** (2017) Software grand exposure: SGX cache attacks are practical. *arXiv e-prints*, page arXiv:1702.07521.
Götzfried J, **Eckert M**, **Schinzel S and Müller T** (2017) Cache attacks on intel SGX. In *Proceedings of the 10th European Workshop on Systems Security*, EuroSec'17. New York: Association for Computing Machinery.
Nilsson A, **Bideh PN and Brorsson J** (2020) A Survey of Published Attacks on Intel SGX.
UK Companies Act (2006).
UK Consumer Rights Act (2015).
UK Human Rights Act (1998).
UK Unfair Contract Terms Act (1977).
van Schaik S, **Kwong A**, **Genkin D and Yarom Y** (2020) SGAxe: How SGX Fails in Practice. https://sgaxeattack.com/.
Writer D (2020) DigixDAO's Dissolution. https://medium.com/digix/digixdaos-dissolution-655be3110196.