






Plateau: A Secure and Scalable Overlay Network for Large Distributed Trust Applications

John Augustine¹, Wahid Gulzar Bhat¹, and Sandip Nair²

¹ Indian Institute of Technology Madras, Chennai, India
augustine@iitm.ac.in

² Columbia University, New York, USA
sdn2124@columbia.edu

Abstract. We propose a novel two-tiered overlay network design called *plateau*. It has two levels: a small upper-level that regulates entry of new nodes into the network, and a lower-level comprising all nodes. The lower level is a well-connected expander that is ideal for building peer-to-peer distributed trust applications. It is designed to be secure despite the presence of adversarial Byzantine nodes and resilient to large amounts of churn. The good nodes only need to communicate with their neighbors in the network, thus making plateau fully distributed. Membership in the network must be earned through proof-of-work that is verified by the upper-level nodes. Plateau is robust despite heavy churn controlled by an adversary, i.e., up to $C = \text{poly}(n)$ number of nodes can join and leave the network per round without disrupting the network structure; n is the total number of good nodes in the network. As long as the compute power controlled by the Byzantine adversary is bounded, the number of Byzantine nodes in the network is kept in check and, more importantly, they will not be able to disrupt the structure or functioning of the overlay network. Additionally, we show that all resources needed to operate this network is bounded polylogarithmically with respect to n .

1 Introduction

Since the invention of Bitcoin by Satoshi Nakamoto [29], we have seen a significant increase in peer-to-peer distributed trust systems. A large number of cryptocurrencies have sprouted over the years and a tremendous amount of research has been invested in this technology in the last decade. The key innovation in Nakamoto's work that is driving this surge is blockchains, a technology by which a peer-to-peer network can maintain a trustworthy record of transactions. Thus, blockchains appeal to a much wider class of applications that require trust between parties. An important aspect of all of these applications is the large volume at which they are intended to operate essentially catering to large populations at national, continental and global scale.

Current blockchain implementations are unfortunately not built for scale [10]. There are many factors that limit them. While some issues like the energy cost of

consensus have drawn significant attention, others have been largely ignored. The actual peer-to-peer network on which the blockchains operate are surprisingly small when compared to the scope and scale of their applications. For example, despite significant use of bitcoin, the actual number of peers that operate is quite small. As of 2018, measurements show that the number of peers is about 14000 [30]. We posit that this small scale of the peer-to-peer network will be a limiting factor as large countries and economic blocs like USA, EU, China, and India seek to employ public blockchain based distributed trust applications for their citizens. If we contrast this with the penetration of the Internet into households across the globe, we realize that distributed trust applications built on current peer-to-peer networks are a far cry from the scale we need for the applications we wish to build on them.

Much of the research in consensus mechanisms abstract away the network issues by assuming that flooded messages reach most nodes within some time period. Such convenience assumptions are acceptable for small networks that are currently deployed. The widely used approach is to maintain the peer-to-peer network as an unstructured random graph. New nodes that wish to join the network connect to random peers obtained from established seeders who crawl through the network and maintain current a list of peers [8]. Such seeders have two drawbacks. Firstly, while they operate well in small networks, their performance in large systems is more challenging. It will be very hard for seeders to publish a list of current nodes at billion nodes scale. Secondly, there is very little mathematical basis for their guarantees. With large amounts of churn, the data they hold can quickly become stale. This, in turn, can lead to poorly connected or even disconnected networks.

Thus, we need to design peer-to-peer networks that can scale well in practice to reach close to Internet scale. At the same time, given the high stakes, we also require strong guarantees backed by rigorous mathematical proofs. A peer-to-peer network capable of hosting large scale distributed trust applications must reliably and efficiently provide some basic functions and properties. Perhaps, the most important function is efficient information spreading, which requires the network to be well connected with good network expansion [25] and of low diameter. The (vertex) expansion of a network graph $G = (V, E)$ is defined as $\min_{S \subset V, |S| \leq |V|/2} |N(S)|/|S|$, where $N(S)$ is the open neighborhood of S (i.e., excluding S). G is said to be an expander if its expansion is bounded from below by a constant. Creating such expander networks with efficient information spreading properties require fast and reliable sampling of random peer nodes [27]. Sampling is straightforward in small networks because the full list of nodes can be effectively maintained by seeders (as it is currently done). However, when the network becomes large, we will need a more distributed mechanism typically employing random walks [14].

To make matters worse, there are several security challenges. Peer-to-peer networks are permissionless allowing any node to participate – including those that are potentially malicious (also called Byzantine nodes). Such Byzantine nodes can affect the network in many ways. They can create cuts in the net-

work and hinder the flow of information across the cut, thereby causing eclipse attacks [18]. It can also be hard to pin down the true identity of participants because of Sybil attacks whereby multiple IP addresses can be created [16, 17]. Furthermore, a large number of malicious nodes can engage in a denial-of-service attack wherein they target some nodes and send repeated messages that overwhelm those nodes and render them unresponsive.

Finally, any peer-to-peer network must be able to tolerate large amounts of churn and other forms of network dynamics. Studies have shown that up to 97% of the nodes exhibit intermittent network connectivity [19]. Moreover, nodes will only participate as long as there is an immediate benefit to them and will leave when there is none. In fact, it has long been established that up to 50% of the nodes can be renewed within an hour, but the number of active peers does not change dramatically because the number of joins and leaves are about the same within small time frames [37]. It is therefore in the interest of peer-to-peer network designers to allow peers to efficiently join and leave without disrupting the network. Sybil attacks and churn coupled together can be quite damaging because the mechanisms in place to let new nodes join the network must be smart enough to ensure that sybils do not abuse the churn facility.

The key mechanism that researchers have used in order to tackle this combination of malicious behavior and churn is to make the participants pay a price in the form of resource burning [17]. This is a mechanism by which the participants are able to prove that they spent some effort or resource to earn their place in the peer-to-peer network. In fact, Gupta *et al.* [16, 17] argue game theoretically that resource burning is a crucial requirement that cannot be avoided when dealing with malice and churn, a position that we share as well. The most common form of resource burning is a mechanism called proof-of-work where a computational puzzle is solved – typically, one that is hard to solve but easy to verify. It is of course a widely used technique for consensus in bitcoin and other cryptocurrencies. Of course, while we may not be able to avoid resource burning, from a sustainability point of view, it is imperative that we minimize its use. In the rest of the paper, we use the term proof-of-work out of deference to its familiarity, but our ideas will go through under any other reasonable form of resource burning as well. Finally, we note that – as in every other proof-of-work based system – we must limit the computation power of the Byzantine adversary to within a fraction of the computational power vested with good participants.

1.1 Our Contribution

In this work, we have made first steps towards designing a secure peer-to-peer overlay network called *Plateau* that can arguably scale well, handle large amounts of churn, and resilient to Byzantine nodes. Our emphasis is on ensuring that the desired properties can be formally proved. Towards this goal, we empower a single adversary to orchestrate the behavior of Byzantine nodes *and* the nature of churn. We assume that the adversary is vested with $1/4$ fraction of the compute power that good nodes possess. Section 2 provides a detailed description of our model. The only cryptographic tools assumed are private channels between

nodes and a proof-of-work mechanism. We do not assume public key infrastructure. The non-triviality of this work comes from carefully designing network and the maintenance protocols so that the properties described below (and formally stated in Theorem 2) hold with high probability (whp)¹.

Plateau is fully distributed. Nodes need to only be aware of their neighbors' IDs and interact with them. Nevertheless, membership is globally secure in the sense that nodes cannot arbitrarily enter the network. Membership in the network must be earned through proof-of-work that is verified by the nodes in the upper level.

The overlay has low diameter and is sparse with both diameter and degree at most logarithmic in the size of the network. The network graph induced by the good nodes is well-connected in the form of an expander, specifically in the sense that its vertex expansion is lower bounded by a constant. Thus, our network is resistant to eclipse attacks. We exploit this expansion to provide a sampling mechanism based on random walks that is resilient to Byzantine behavior. Furthermore, Plateau is designed with judicious use of proof-of-work that makes it resilient to Sybil and DoS attacks. Plateau is robust despite heavy churn controlled by the adversary. Up to $C \in O(n/\text{polylog}(n))$ nodes can join and leave the network per round without disrupting the network.

Its scalability is highlighted by the fact that all resources used are small compared to the overall size of the network and more importantly competitive with the amount of churn. Communication between nodes is via small messages of $\text{polylog}(n)$ bits. Each node has at most $\text{polylog}(n)$ neighbors at any point in time and only needs to communicate with its neighbors. The total number of messages sent/received by all nodes during any round is at most $\tilde{O}(C)$.

Prior works typically assume that the new nodes are automatically connected to appropriate nodes within the network. This is in stark contrast to reality where new nodes must depend on information provided by seeders [8]. Our work formally includes this aspect in that our protocol requires a dynamic whiteboard² with $\tilde{O}(C)$ bits that is visible to any new node that seeks to join the network. Each new node samples (from the whiteboard) a random $\text{polylog}(n)$ sized chunk of information that includes a suitable proof-of-work puzzle that the new node must solve. It also includes IDs of the appropriate nodes within the network that it must connect to and submit the solution to the puzzle in order to gain entry into the network. Our protocol updates the whiteboard at the rate of $\tilde{O}(C^2/n)$

¹ We say that an event E holds with high probability (whp) if $\text{Prob}[E] \geq 1 - 1/n^\eta$ for any fixed parameter η that is independent of n , but may depend on constants used in the algorithm.

² We use the term whiteboard to abstract out the ability to expose information about the network to the world. This is a crucial requirement for any network to handle churn. Otherwise, new nodes will not know where to connect. In current cryptocurrency systems like Bitcoin, we have specialized servers called *seeders* that provide this service [8]. Other alternatives include using the blockchain itself to expose this information [1]. The main design issue is to ensure that the whiteboard only needs to store a bounded amount of information and that updates to the whiteboard are not too fast.

bits per round in order to keep up with the dynamic updates within the network. Thus, as long as the churn rate is $\tilde{O}(\sqrt{n})$, the update rate is at most $\text{polylog}(n)$. When C is larger, the whiteboard must be updated at a commensurately larger rate. This is essentially the best we can do because we prove a matching lower bound (within $\text{polylog}(n)$ factor) for the update rate.

Importantly, our work assumes that an adversary controls the behavior of all Byzantine nodes including when to seek membership, when to exit, whether to send messages, and what messages to send. The adversary controls churn amongst good nodes in the following oblivious manner that models the worst case (but not malicious) behavior. At the time when a new good node enters the network, the adversary decides how long it will stay in the network. This choice must not violate the churn rate C . Specifically, the adversary is not allowed to churn out more than C nodes per round. Thus, the adversary can impose worst case patterns by which good nodes can churn in and out, but it cannot maliciously and/or adaptively decide when to churn out good nodes.

The main novelty in plateau’s design is its two levels. The upper-level is smaller and commensurate in size with the rate at which new nodes join. It consists nodes that act as juries and regulate entry into the network. The lower-level is the essential peer-to-peer (P2P) network that is scalable to large sizes. It is well-connected with good expansion (thereby allowing us to spread information fast and also sample random nodes via random walks) making it ideal for building peer-to-peer distributed trust applications. We show that Plateau can be maintained despite the Byzantine adversary possessing up to a fixed $\beta < 1/4$ of the computational power possessed by good nodes.

1.2 Related Works

In the early years of P2P networks, several prominent overlay network designs like Chord [36], CAN [34], Pastry [35], Tapestry [39] were proposed. Following those early proposals, there has been extensive research on designing robust overlay networks with a variety of useful and rigorously proved characteristics like well-connectedness, low diameter, expansion, low degree, and robustness to network churn and malicious behaviour [3–5, 9, 11, 20, 21, 24–26, 31–33]. For our purpose, we will highlight a few works that are relevant to our goals and design principles of maintaining large scale well connected overlay networks that are robust against Byzantine behavior and adversarial churn.

One of the earliest works in this regard was by Fiat and Saia [13] where items can be stored in a network and most items can be retrieved efficiently despite an adversarial removal of a large fraction of the nodes. In fact, their solution can be adapted to situations where the adversary takes control of a fraction of the nodes (not just remove them). Unfortunately, it is unclear how their overlay can be maintained in the presence of heavy churn. More recently, Guerroui *et al.* [15] presented a Byzantine resilient overlay maintenance protocol called Neighbors on Watch (NOW) that bears significant resemblance to our protocol. They also maintain an expander graph on supernodes (containing $\Theta(\log n)$ peer nodes) and ensure random distribution of peers within the supernodes. They show how a

new node can join or an old node can leave. To the best of our understanding, their design and analysis is limited to just a few nodes (up to $O(\log n)$ nodes) joining or leaving at a time. To their credit, they employ a much stronger form of adversary that can churn out any choice of nodes at any time.

Several gossip based sampling protocols have been studied in the past [9, 22, 23]. The work by Bortnikov *et al.* [9] is quite relevant to ours. It has two components: the sampling component and the gossiping component. The sampling component maintains a list of uniform samples from the set of IDs that passed through the node. The gossiping component spreads IDs across the network and maintains the dynamic view of the system. There are, however, two significant drawbacks. Firstly, the protocol requires each node to store $\Theta(n^{1/3})$ IDs locally. Secondly, the analysis of convergence to uniform random samples holds only when the churn ceases, which is unfortunately not the case in the real world. Jesi *et al.* [22] provide a Byzantine resilient peer sampling mechanism that employs identifying and blacklisting nodes that behave maliciously. Johansen *et al.* [23] provide a robust pseudorandom structure that is useful for good nodes to maintain correct membership views. Their work maintains a complete membership view, which is unfortunately unscalable for very large networks.

Peer-to-peer networks, as we have mentioned before, experience heavy network churn [19, 37]. Quite a bit of research has gone into designing overlays that are resilient to heavy churn [3, 4, 6, 11]. Awerbuch and Scheideler [6] employed the cuckoo rule by which new nodes can join with minimum displacement of existing nodes. An interesting deterministic P2P overlay network was proposed by Kuhn *et al.* [24], but the price of determinism is that their approach only works with a very small rate of joins and leaves. Augustine *et al.* [3] show how to maintain an overlay network with good expansion despite heavy churn. They employ random walks to sample random nodes and place new nodes in random locations in order to maintain good expansion. Drees *et al.* [11] design an overlay network that can handle heavy adversarial churn, but their model requires nodes to join and leave gracefully with a forewarning of at least $\Omega(\log \log n)$ rounds. Augustine and Sivasubramanian [4] provide an overlay design called Spartan that has many similarities to our approach. Both [24] and [4] employ supernodes (or committees) of size $\Theta(\log n)$ nodes. The major drawback of all these works is that they are not shown to be resilient to Byzantine failures.

In a recent work [1], Aradhya *et al.* show how to maintain a Byzantine resilient blockchain overlay network using the blockchain itself as a means to share information among the peers. This work bears many common features with ours. They also show how the network can tolerate churn. While our work uses arbitrary expander graph structure, they use a hypercubic network structure for the overlay. Their work is specific to blockchain systems, but our work is more general and applicable to any secure peer-to-peer network.

Organization. We begin with a formal description of the model in Sect. 2 and also describe a few important tools that we use in our design. We then present a detailed description of Plateau’s design in Sect. 3. Proofs and pseudocode have not been included due to insufficient space.

2 Model and Preliminaries

We begin with a formal description of our network model. Our goal is to design a sustainable peer-to-peer overlay network that can serve as a platform for building large scale distributed trust applications. See Fig. 1 for a schematic. We use the term *node* to refer to the peers that participate in the system. Some of these nodes will be Byzantine (i.e., malicious) while others are good. Moreover, network must also tolerate churn whereby nodes can join and leave. The *System* comprises both the network and all the nodes (both Byzantine and good) that are actively seeking membership within the network. For simplicity, we assume that the number of good nodes n in the system at any point in time is stable.

For simplicity, we assume that the system operates synchronously with rounds being the basic unit of time. Due to churn, up to C good nodes, for some $C \in [0, n/\text{polylog}(n)]$, can leave the system per round and an equal number³ must enter the system per round in order to maintain a stable number of good nodes in the system. When a node enters the system, it must be integrated into the network by a protocol that maintains the network (and this may take some time). The nodes in the system, but not yet integrated into the network are called *seekers* because they are nodes seeking membership within the network.

We assume that each good node has a unique ID – typically its IP address – that can be used both to uniquely identify it as well as to form network connections. Moreover, each good node is capable of a bounded amount of computational work (or just work). The Byzantine nodes are controlled by a single Byzantine adversary that can create as many Byzantine nodes as it needs, but the overall computational power of the Byzantine adversary is limited to a positive fraction $\beta < 1/4$ (known as the *Byzantine power parameter*) of the total compute power of good nodes.

The goal is to design a network that is robust despite churn and Byzantine nodes. In particular, the good nodes must maintain a degree of at most $O(\log n)$ and must induce an expander graph with vertex expansion bounded from below by a constant. The specific network we present is called the *Plateau network* (or just network) and for this reason, we refer to the system as the *Plateau system*. Plateau must ensure that, at any point in time, all but $O(C)$ good seekers are integrated into the network. It is inevitable that the network may have integrated some Byzantine nodes as well but we wish to ensure that they are at most β^*n at any point in time for some fixed fraction $\beta^* < 1/2$. Moreover, those Byzantine nodes must be incapable of compromising the guar-

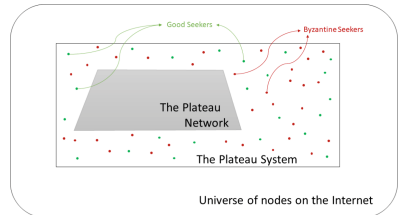


Fig. 1. The Plateau System and the Plateau network.

³ Our design is sufficiently robust to admit variation between the number of nodes joining and leaving as long as the total number of good nodes stays bounded within some reasonable $\Theta(n)$.

antees offered by the network. The term *Plateau system* (or just *system*) denotes the network and the seekers; see Fig. 1 for a schematic of the system and the network.

Churn Model. We now describe the churn process in bit more detail. A node that is neither a member nor a seeker is called an *external node* and such an external node can become a seeker at any time; we call this *churning in*. Likewise, a node in the system (regardless of whether it is a member or a seeker) can leave the system and become an external node; we call this *churning out*. The Byzantine adversary controls nodes churning in and out subject to the following constraints.

At most C good nodes can be *churned in* per round and an equal number churned out. Any number of Byzantine nodes can be churned in and churned out. An *epoch* is defined to be n/C rounds and corresponds to the time required by an adversary to completely replace the current set of nodes with a new set of nodes. The network must ensure that good seekers gain entry into the network in a timely fashion through a process called *integration*. Specifically, we wish to ensure that the number of good seekers is no more than $O(C)$ at any time (whp). Moreover, every good seeker node should be integrated into the network within $O(\log n)$ rounds (whp).

Nodes can be *churned out* either by the protocol or by the adversary. When all good nodes drop their connections with a node u , then it is considered churned out by the protocol (as long as it is clear that good nodes cannot be tricked into forming connections with u later on). Such churn outs are expected to happen when a node is unable to provide proof of work that the protocol may require of it from time to time. The protocol must be designed to ensure that good nodes are not churned out in this manner because they are expected to be willing to spend one unit of computational power per epoch. The time when a good node v is churned out by the adversary must be specified when the node is churned in. (A good node will not be aware of its churn out time.) There is no incentive for the adversary to actively churn out Byzantine nodes. However, since the computational power of the adversary is bounded, Byzantine nodes that are unable to provide proof-of-work must be churned out by the protocol.

Communication Model. Nodes can communicate with each other in one of two modes: either through established overlay links (e.g., TCP sessions) or through ports that are open. Formation of an overlay link between two nodes u and v must be initiated by one node and consented by the other; such a link can be formed in one round. We assume that each node can maintain $O(\log n)$ overlay links. Alternatively, each node has $O(\log n)$ ports numbered $\{1, 2, \dots, O(\log n)\}$ through which it can listen for new messages or new connections. Thus, if a node u knows the ID of node v , then u can send v a message through some port x . The message will be delivered to v if no other node is also attempting to send a message to v through x at the same time. Messages will be dropped when such conflicts occur. We assume that u will be aware of whether the message reached v or not. We require each message (sent through either mode) to be small in size, i.e., at most $O(\text{polylog } n)$ bits. Furthermore, We wish to ensure that the

total number of messages sent by good nodes are at most $M \in O(C \log n)$ per round. We similarly limit the number of messages sent by Byzantine nodes to also be within the same limit M .

Public Whiteboard. To facilitate integration, the network is allowed to publish information on a *whiteboard* that is available for public viewing. For the purpose of this paper, we abstract away the details of how such a whiteboard may be implemented. The whiteboard, however, must be limited to displaying $\tilde{O}(C)$ bits of information that is updated at the rate of $o(C)$ bits per round, i.e., at most $o(C)$ bits can be erased and at most $o(C)$ bits can be written per round.

Proof-of-Work. We assume that proof-of-work puzzles can be solved by expending one unit of compute power⁴. To solve such a puzzle, a node u requires an input bit string r and its own ID and computes a nonce bit string q such that $h(r|ID(u)|q)$ has sufficiently many leading zeros, where h is a random oracle hash function. Each good node must be willing to spend 1 unit of computational power for integration and subsequently spend one unit of computational power per epoch. The number of rounds required to solve one proof of work puzzle is assumed to be within $O(\log n)$ rounds. The total computational power of the Byzantine adversary is assumed to be βn per epoch, where $\beta > 0$ known as the *Byzantine power parameter* is a fixed constant bounded strictly below $1/4$. I.e., the Byzantine nodes, in total, can solve $n/4$ proof-of-work puzzles per epoch. Additionally, whenever a good node is churned in, the Byzantine adversary is credited with β units of computational power that must be spent within $O(\log n)$ rounds. This is to ensure that the Byzantine adversary is empowered to churn in Byzantine nodes into the network.

Useful Tools and Techniques. We use several standard tools and techniques that we explain in greater detail in the full version. We rely on expander graphs [38] for fast mixing time, low diameter (i.e., both logarithmic in the size of the network) and established tools for creating and maintaining them in dynamic environments [3,32]. Furthermore, we assume that Byzantine agreement [12] and collective coin tossing [28] can be executed $O(\log n)$ rounds whp.

3 The Plateau Network Design and Statement of Results

We now describe our proposed network design. It relies crucially on sets of $\Theta(\log n)$ nodes called *supernodes* that are interconnected to form the Plateau network. The supernodes partition the set of nodes, thus there are $n/c \log n$ supernodes for some sufficiently large constant c . We say that a supernode is *b-Byzantine-Bounded* for some $b \in [0, 1]$ if *fewer than* b fraction of the nodes in it are Byzantine. The network is said to be *b-Byzantine-Bounded* if all supernodes in it are *b-Byzantine-Bounded*. Our goal is to guarantee that the Plateau network is $(1/3)$ -Byzantine-Bounded (i.e., every supernode is $(1/3)$ -Byzantine-Bounded).

⁴ This is a simplifying assumption. We can also model the compute power required to solve a puzzle as an exponential random variable.

Thanks to this limited influence by the Byzantine adversary, supernodes can serve as committees that decisively act by invoking Byzantine Agreement [12].

The list of nodes in a supernode s is maintained as common knowledge among all nodes in s . Thus, whenever all good nodes in s unanimously propose a value, they can initiate Byzantine Agreement and ensure that s (as a single entity) will be able to decide on one of those values. Moreover, each supernode can execute Micali and Rabin’s unbiased coin tossing protocol and generate unbiased random bits that all good nodes within s agree upon.

Two supernodes s_1 and s_2 are said to be connected by a *logical link* if every good node in s_1 (resp., s_2) is aware of all members in s_2 (resp., s_1) and has successfully established an overlay link with every good node in s_2 (resp., s_1).

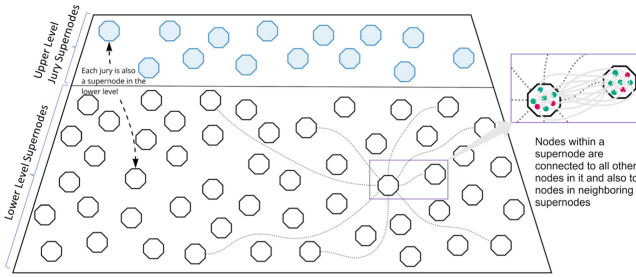


Fig. 2. The Plateau network architecture comprising (1/3)-Byzantine-Bounded supernodes in two levels. Each jury supernode is also a supernode in the lower level. The supernodes in the lower level (resp., juries in the upper level) are connected via logical links (not shown) to form an expander graph G (resp., H).

The Two-Tier Plateau Structure. As mentioned before, Plateau comprises two levels: the lower level comprising the set of all supernodes S and the upper level comprising a (dynamic) set of $\Theta(M)$ supernodes J called juries. Note that juries will have to continue their role in the lower level even while serving as juries. See Fig. 2 for a schematic of the Plateau architecture.

The supernodes at both levels are connected via logical links in the form of (constant degree) expander graphs: $G = (S, E)$ for the lower level and $H = (J, E')$ for the upper level. Our design therefore ensures that each supernode has established logical links to at most $O(1)$ other supernodes. Thus, the number of overlay links at each individual node is at most $O(\log n)$.

Secure Messages. Two supernodes s_1 and s_2 that are connected by a logical link can communicate with each other at will via *secure messages* (explained shortly) with the twin security guarantees of authentication and integrity. When the (good) nodes in (say) s_1 wish to send a secure message to the nodes in s_2 , they individually send the same message to every node in s_2 . At the receiving end, i.e., at s_2 , the good nodes accept all messages sent by at least a 2/3 fraction of the nodes in s_1 . Any message sent by fewer than a 2/3 fraction is discarded. Such

a message sent by all good nodes in s_1 to all good nodes in s_2 in this coordinated manner is deemed a secure message. Notice that s_2 receives a secure message from s_1 iff s_1 sends a secure message to s_2 . Thus, as long as all the good nodes in s_1 are agreed on what message to send, the recipient s_2 knows that the message indeed originated at s_1 (i.e., the sender’s authenticity is guaranteed) and that the message has not been tampered with (i.e., its integrity is guaranteed). Notice however that these secure messages are not guaranteed to be private. If either s_1 or s_2 has even one Byzantine node, the Byzantine adversary will be able to learn the contents of the secure message.

A node u cannot become a member of a supernode without proper credentials. If u is a seeker and has performed the requisite proof-of-work, a jury supernode will admit it into the network and move u to a random supernode s via a secure random walk (described shortly). Importantly, u cannot influence the choice of s . Subsequently u may be moved around via secure random walks roughly once every epoch. Thus, there is no provision for a node u to join an arbitrary supernode. Node u can only join a supernode s as a consequence of secure random walks that explicitly introduce u to s .

Plateau Maintenance. To maintain Plateau, we operate in *maintenance cycles* of $m \in \Theta(\log n)$ rounds. In each cycle, $\tilde{\Theta}(1 + \frac{C^2 \log^3 n}{n})$ juries are replaced by randomly chosen supernodes from the lower level and an expected C nodes are moved to random locations. Simultaneously, new nodes that wish to join are integrated into Plateau after proper vetting of their proof of work.

Replacement of Juries. The juries $J \subset S$ (chosen uniformly at random) regulate the entry of new nodes into the network. We rotate in new jury supernodes during each maintenance cycle and simultaneously evict an equal number from the upper level. All the IDs of nodes in the newly inducted juries are added to the whiteboard and the IDs of nodes in the evicted juries are deleted from the whiteboard. The rotation of juries ensures that the list of nodes in juries written to the whiteboard are sufficiently current.

Let $\mathbf{r} = \tilde{\Theta}(1 + \frac{C^2 \log^3 n}{n})$ denote the *refresh rate*, i.e., the rate at which juries are rotated in and out of the upper level per round. At the start of every maintenance cycle, the protocol picks $\mathbf{r}m \in O(\mathbf{r} \log n)$ random juries j and marks them for replacement. Simultaneously, an equal number of random lower level supernodes s are called for jury duty and are installed in H , with each s in the same neighborhood of a corresponding j in H ; juries marked for deletion can now be deleted from H . Thus, the topology of H remains stable, but its vertices are rotated in and out regularly. Note that the random choices of j and s can be made via secure random walks of length $\Theta(\log n)$ performed on H and G , respectively. The full version contains formal pseudocode.

Information Published on the Whiteboard. The whiteboard maintains a current list of juries and the constituent nodes within those juries (including Byzantine nodes). It also includes a random bit string r that is updated every cycle. The whiteboard will only accept updates given by secure messages from current juries. Whenever a jury j leaves the upper level, it informs the whiteboard

and is erased from the whiteboard. When a new jury j enters the upper level, a pre-existing jury j' (typically a neighbor of j in H) must inform the whiteboard so that j and its constituent members can be included in the whiteboard. We also maintain a designated lead jury j^* that generates a random bit string r in each cycle and updates the whiteboard with r (while the earlier string is erased).

Reassignment of Nodes. Byzantine nodes can selectively sever ties with nodes both within its own supernode as well as neighboring supernodes. Moreover, we must ensure that Byzantine nodes don't freeloader or selectively stagnate and pile up in some supernode. To avoid these issues, each node u in each supernode s is reassigned once every (expected) n/mC cycles to a new supernode s' chosen randomly through a secure random walk. The choice of u is via collective coin tossing by nodes within s such that the time between two consecutive reassignments for u is geometrically distributed with $p = mC/n$. When chosen for reassignment, u must first show proof of work requiring a one unit of computation. Then the nodes in s vote on whether (i) u has shown the correct proof of work *and* (ii) has correctly executed all protocols during the last epoch and perform a Byzantine agreement to decide whether to retain u or churn it out. If the agreement is not in favor of u , all good nodes in s will sever their links with u and also inform all neighboring supernodes through secure messages, thereby effectively churning out u . If u survives, it is forced to make a secure random walk for $\ell_G \in \Theta(\log n)$ steps where u will be chaperoned to a new random supernode s' .

Integrating Seekers into the Network. Each seeker x reads the current random string r from the whiteboard and solves the puzzle pertaining to $(r|ID(x))$. The solution is a nonce bit string t such that $h(r|ID(x)|t)$ has at least ℓ leading zeros for some predefined ℓ and a commonly agreed random oracle hash function h . This requires a 1 unit of compute power and time that is at most $O(\log n)$ rounds. The seeker x then picks a random jury j and sends its proof of work to every node in j (listed in the whiteboard) through randomly chosen ports. If more than half of the members of j receive the proof and acknowledge it, then x sends an accept message to nodes in j and waits for j to integrate x into the network. Otherwise, x sends a reject message to nodes in j and repeats the process with a new random jury. The juries wait for seekers to send proof and acknowledge them. When a seeker x sends an accept, the jury begins Byzantine agreement to either approve or reject the request. If approved, a secure random walk is initiated and x is chaperoned to a random supernode in G .

Our results are formalized by the following two theorems.

Theorem 1. *Any whiteboard based P2P network (with whiteboard size $O(C)$) that experiences churn at the rate of C nodes per round must update the whiteboard at the rate of $\tilde{\Omega}(C^2/n)$ bits per round.*

Theorem 2. *The Plateau system is designed with the following guarantees that hold with high probability as long as the Byzantine power parameter β is a fixed constant that is bounded strictly below $1/4$ and the churn rate $C \in [0, n/\text{polylog}(n)]$.*

Byzantine Boundedness. *The Plateau network will be $(1/3)$ -Byzantine-Bounded for at least $T \in \Omega(n^k)$ rounds for fixed k .*

Network Properties. *The network induced by the good nodes within the Plateau network forms an expander with vertex expansion bounded from below by a constant. Thus, its diameter is $O(\log n)$. Moreover, the number of overlay edges incident to any good node is at most $O(\log n)$.*

Quick Integration. *Seekers will integrate within $O(1)$ rounds on expectation and the expected number of seekers waiting to be integrated will be at most $O(C)$ at any time.*

Efficient Whiteboard. *The whiteboard employed by Plateau is of size at most $\tilde{O}(C)$ and is updated at the rate of $\tilde{O}(C^2/n) \in o(C)$ bits per round when C is at most $n/\text{polylog}(n)$. In fact, the update rate is at most $\tilde{O}(1)$ if $C \in \tilde{O}(\sqrt{n})$ and this is optimal to within a $\text{polylog}(n)$ factor.*

4 Concluding Remarks and Future Work

We have presented a P2P network architecture called Plateau that is able to regulate the entry and exit of nodes even at high churn rates. Our design is quite generic and can be easily adapted in a variety of ways. Our choice of expander graph structure is in keeping with the long line of works on P2P networks that rely on expansion [15, 25, 31]. Moreover, it closely resembles the P2P networks we see in practice and they are known to be robust even under adversarial deletions [7]. However, expander graphs can be replaced by other structures that have good sampling properties (e.g., hypercubes [2] and butterflies) with potential benefits. For example, the Spartan structure [4] that is based on the butterfly network facilitates addressable supernodes and efficient routing between them. This can be used to build distributed hash tables.

Furthermore, for simplicity, we assumed that the number of good nodes is stable at n . However, we can easily adapt Plateau’s design to varying values of n . This can be done very robustly when the rate of change of n is $\text{polylog}(n)$ per round by adapting G using [32]. For more dramatic changes, we can use a more structured approach wherein G is a hypercube or a butterfly. When n increases or decreases dramatically, such structures can be expanded or contracted by incrementing or decrementing their dimension using ideas from [2].

We believe that a thorough simulation of Plateau will greatly help in understanding its viability in practice. Moreover, the current paper is limited to synchronous systems. Extending these ideas to asynchronous systems is an important next step. Finally, the current work abstracts away the details pertaining to implementing a whiteboard, but these details need to be worked out for Plateau to work in practice.

Acknowledgements. We thank Seth Gilbert and Aquinas Hobor for preliminary discussions in which they suggested the idea of using a whiteboard to facilitate new nodes joining the network. The first author was supported in part by Extra-Mural Research Grant (file number EMR/2016/003016) and MATRICS grant (file number MTR/2018/001198). He is currently supported by the potential Centre of Excellence

in Cryptography Cybersecurity and Distributed Trust (CCD) under the IIT Madras Institute of Eminence scheme. The third author worked on this project as an intern at IIT Madras supported by an Information Security Education and Awareness (<https://isea.gov.in/>) Phase II project (CCECEP22VK&CPCSE1415).

References

1. Aradhya, V., Gilbert, S., Hobor, A.: OverChain: building a robust overlay with a blockchain (2022). <https://arxiv.org/abs/2201.12809>
2. Augustine, J., Chatterjee, S., Pandurangan, G.: A fully-distributed scalable peer-to-peer protocol for Byzantine-resilient distributed hash tables. In: SPAA, pp. 87–98 (2022)
3. Augustine, J., Pandurangan, G., Robinson, P., Roche, S.T., Upfal, E.: Enabling robust and efficient distributed computation in dynamic peer-to-peer networks. In: FOCS (2015)
4. Augustine, J., Sivasubramaniam, S.: Spartan: a framework for sparse robust addressable networks. In: 2018 International Parallel and Distributed Processing Symposium (IPDPS), pp. 1060–1069 (2018)
5. Awerbuch, B., Scheideler, C.: The hyperring: a low-congestion deterministic data structure for distributed environments. In: SODA (2004)
6. Awerbuch, B., Scheideler, C.: Towards a scalable and robust DHT. *Theory Comput. Syst.* **45**(2), 234–260 (2009)
7. Bagchi, A., Bhargava, A., Chaudhary, A., Eppstein, D., Scheideler, C.: The effect of faults on network expansion. *Theory Comput. Syst.* **39**(6), 903–928 (2006)
8. Bitcoin P2P network official documentation. https://developer.bitcoin.org/devguide/p2p_network.html. Accessed 25 Apr 2022
9. Bortnikov, E., Gurevich, M., Keidar, I., Kliot, G., Shraer, A.: Brahms: Byzantine resilient random membership sampling. *Comput. Netw.* **53**(13), 2340–2359 (2009)
10. Croman, K., et al.: On scaling decentralized blockchains. In: Clark, J., Meiklejohn, S., Ryan, P.Y.A., Wallach, D., Brenner, M., Rohloff, K. (eds.) FC 2016. LNCS, vol. 9604, pp. 106–125. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53357-4_8
11. Drees, M., Gmyr, R., Scheideler, C.: Churn-and DoS-resistant overlay networks based on network reconfiguration. In: SPAA 2016, pp. 417–427. ACM (2016)
12. Feldman, P., Micali, S.: An optimal probabilistic protocol for synchronous Byzantine agreement. *SIAM J. Comput.* **26**(4), 873–933 (1997)
13. Fiat, A., Saia, J.: Censorship resistant peer-to-peer networks. *Theory Comput.* **3**(1), 1–23 (2007). <https://doi.org/10.4086/toc.2007.v003a001>. <https://www.theoryofcomputing.org/articles/v003a001>
14. Gkantsidis, C., Mihail, M., Saberi, A.: Random walks in peer-to-peer networks: algorithms and evaluation. *Perform. Eval.* **63**(3), 241–263 (2006). *P2P Computing Systems*
15. Guerraoui, R., Huc, F., Kermarrec, A.M.: Highly dynamic distributed computing with Byzantine failures. In: PODC 2013 (2013)
16. Gupta, D., Saia, J., Young, M.: Resource burning for permissionless systems (invited paper). In: Richa, A.W., Scheideler, C. (eds.) SIROCCO 2020. LNCS, vol. 12156, pp. 19–44. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-54921-3_2
17. Gupta, D., Saia, J., Young, M.: Bankrupting sybil despite churn. In: ICDCS, pp. 425–437 (2021)

18. Heilman, E., Kendler, A., Zohar, A., Goldberg, S.: Eclipse attacks on bitcoin's peer-to-peer network. In: 24th USENIX Security Symposium (USENIX Security 2015) (2015)
19. Imtiaz, M.A., Starobinski, D., Trachtenberg, A., Younis, N.: Churn in the bitcoin network: characterization and impact. In: 2019 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), pp. 431–439 (2019)
20. Jacob, R., Richa, A., Scheideler, C., Schmid, S., Täubig, H.: SKIP+: a self-stabilizing skip graph. *J. ACM* **61**(6), 36:1–36:26 (2014)
21. Jacobs, T., Pandurangan, G.: Stochastic analysis of a churn-tolerant structured peer-to-peer scheme. *Peer-to-Peer Netw. Appl.* **6**(1) (2013)
22. Jesi, G.P., Montresor, A., van Steen, M.: Secure peer sampling. *Comput. Netw.* **54**(12), 2086–2098 (2010)
23. Johansen, H.D., Renesse, R.V., Vigfusson, Y., Johansen, D.: Fireflies: a secure and scalable membership and gossip service. *ACM Trans. Comput. Syst.* **33**(2) (2015)
24. Kuhn, F., Schmid, S., Wattenhofer, R.: Towards worst-case churn resistant peer-to-peer systems. *Distrib. Comput.* **22**(4), 249–267 (2010)
25. Law, C., Siu, K.Y.: Distributed construction of random expander networks. In: IEEE INFOCOM 2003, vol. 3, pp. 2133–2143 (2003)
26. Mahlmann, P., Schindelbauer, C.: Peer-to-peer networks based on random transformations of connected regular undirected graphs. In: SPAA, pp. 155–164 (2005)
27. Mao, Y., Deb, S., Venkatakrisnan, S.B., Kannan, S., Srinivasan, K.: Perigee: efficient peer-to-peer network design for blockchains. In: PODC 2020, pp. 428–437 (2020)
28. Micali, S., Rabin, T.: Collective coin tossing without assumptions nor broadcasting. In: Menezes, A.J., Vanstone, S.A. (eds.) CRYPTO 1990. LNCS, vol. 537, pp. 253–266. Springer, Heidelberg (1991). https://doi.org/10.1007/3-540-38424-3_18
29. Nakamoto, S.: Bitcoin: a peer-to-peer electronic cash system (2009)
30. Neudecker, T.: Characterization of the bitcoin peer-to-peer network (2015–2018). Technical report. 1, Karlsruher Institut für Technologie (KIT) (2019)
31. Pandurangan, G., Raghavan, P., Upfal, E.: Building low-diameter peer-to-peer networks. *IEEE J. Sel. Areas Commun.* **21**(6), 995–1002 (2003)
32. Pandurangan, G., Robinson, P., Trehan, A.: DEX: self-healing expanders. *Distrib. Comput.* **29**(3), 163–185 (2016)
33. Pandurangan, G., Trehan, A.: Xheal: a localized self-healing algorithm using expanders. *Distrib. Comput.* **27**(1), 39–54 (2014)
34. Ratnasamy, S., Francis, P., Handley, M., Karp, R., Shenker, S.: A scalable content-addressable network. *Comput. Commun. Rev.* **31**(4), 161–172 (2001)
35. Rowstron, A., Druschel, P.: Pastry: scalable, decentralized object location, and routing for large-scale peer-to-peer systems. In: Guerraoui, R. (ed.) Middleware 2001. LNCS, vol. 2218, pp. 329–350. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-45518-3_18
36. Stoica, I., Morris, R., Karger, D., Kaashoek, M.F., Balakrishnan, H.: Chord: a scalable peer-to-peer lookup service for internet applications. *Comput. Commun. Rev.* **31**(4), 149–160 (2001)
37. Stutzbach, D., Rejaie, R.: Understanding churn in peer-to-peer networks. In: SIGCOMM, New York, NY, USA (2006)
38. Vadhan, S.P.: Pseudorandomness. *Found. Trends® Theor. Comput. Sci.* **7**(1–3), 1–336 (2012)
39. Zhao, B.Y., Kubiatowicz, J., Joseph, A.D.: Tapestry: a fault-tolerant wide-area application infrastructure. *Comput. Commun. Rev.* **32**(1), 81 (2002)